Variable Geometry Scramjet Combustor Cavity Multi-Dimensional Treatise for Performance Analysis

Andrew L. Sorensen

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science in Aerospace Engineering

Jonathan T. Black, Co-chair
Daniel D. Doyle, Co-chair
John N. Gilbert
Russell M. Cummings

September 23, 2021
Blacksburg, Virginia

Keywords: Hypersonics, Morphing, Combustor, HiFiRE, CReSS

Copyright 2021, Andrew L. Sorensen
Variable Geometry Scramjet Combustor Cavity Multi-Dimensional Treatise for Performance Analysis

Andrew L. Sorensen

(ABSTRACT)

The abilities of Scramjets and Ramjets, in their respective operating ranges, are partially bridged by dual-mode Scramjets. The limitations of operation are due to making a static motor that is designed to function in both modes resulting in low and high speed restrictions. This study covers the analysis into the ability of morphing the combustor in a Scramjet to allow for expanded operational capacities through simple mechanisms. Through the restriction and expansion of combustor cavity volume, operational capabilities of the engine can, therefore, be modified to best match scenario requirements. Due to the engine's ability to match a wide variety of scenarios the limitations seen in that of the dual-mode Scramjet are avoided through the usage of a morphing combustor. From initial findings using the quasi-1D Canonical REactor Scramjet Simulation (CReSS) solver, progress was made to confirm results through the usage of Computational Fluid Dynamics (CFD). Prior analysis of the momentum balance between stages two and four of the simulated Scramjet engines, the results showed that the variable geometry matched or outperformed the baseline HiFiRE geometry. The analysis revealed points of Mach and altitude where certain combustor volumes demonstrated greater performance. This greater performance is only gained by the ability to tune the engine in flight to react to external factors as there is no dominant geometry for a given range of Machs and altitudes. This tuning allows for the usage of performance mapping to extract the greatest performance possible over a variety of conditions. Further, it allows for the project to be continuously expanded into mapping appropriate reactions to
other initial conditions and stimuli. Using CFD modeling to perform a parametric study on the prior work allows for finer control and analysis of said initial conditions and the resulting flow paths in the variety of tested combustor volumes. From this a discussion is made in regards to the effectiveness of the prior CReSS based analysis of the novel approach.
Variable Geometry Scramjet Combustor Cavity Multi-Dimensional Treatise for Performance Analysis

Andrew L. Sorensen

(GENERAL AUDIENCE ABSTRACT)

The abilities of Scramjets and Ramjets (engines which contain no moving parts as the compression of the incoming air is accomplished by the speed at which they operate with the separating factor being that the scramjets internal flow does not go below supersonic speeds), in their respective operating ranges, are partially bridged by dual-mode Scramjets. Dual-mode Scramjets are scramjets which can function with both sub- and super-sonic internal flow speeds. This being below or above Mach 1 (343 m/s, 767.3 mph) respectively. The limitations of operation are due to making a static motor where the geometry does not change that is designed to function in both modes resulting in low and high speed restrictions. This study continues the analysis into the ability of morphing the combustor, the volume in which the air fuel mixture combests, in a Scramjet to allow for expanded operational capacities through simple mechanisms. Through the restriction and expansion of combustor volume, operational capabilities of the engine can, therefore, be modified to best match scenario requirements. Due to the engine’s ability to match a wide variety of scenarios the limitations seen in that of the dual-mode Scramjet are avoided through the usage of a morphing combustor where morphing in this case is a simple volume change equivalent to that of a slide whistle. From initial findings using the quasi-1D Canonical REactor Scramjet Simulation (CReSS) solver, progress was made to confirm results through the usage of Computational Fluid Dynamics (CFD). Prior analysis of the momentum balance between stages two and four of the simulated Scramjet engines, the results showed that the variable
geometry matched or outperformed the baseline HiFiRE geometry. The analysis revealed points of Mach and altitude where certain combustor volumes demonstrated greater performance. This greater performance is only gained by the ability to tune the engine in flight to react to external factors as there is no dominant geometry for a given range of Machs and altitudes. This tuning allows for the usage of performance mapping to extract the greatest performance possible over a variety of conditions. Further, it allows for the project to be continuously expanded into mapping appropriate reactions to other initial conditions and stimuli. Using CFD modeling to perform a parametric study on the prior work allows for finer control and analysis of said initial conditions and the resulting flow paths in the variety of tested combustor volumes. From this a discussion is made in regards to the effectiveness of the prior CReSS based analysis of the novel approach.
Dedication

To my family and friends who have done far more than their fair share helping me through thick and thin. Thank you.
Acknowledgments

The author acknowledges the Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper. URL: http://www.arc.vt.edu

The author acknowledges the Ted and Karyn Hume Center for National Security and Technology at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper. URL: https://hume.vt.edu

This work was supported in part by high-performance computer time and resources from the DoD High Performance Computing Modernization Program.

The author acknowledges the Hypersonic Vehicle Simulations Institute, sponsored by the Department of Defense High Performance Computing Modernization Program, for supplying technical resources and insights, along with other multi-faceted points of support that have deeply contributed to the results reported within this paper.

The author acknowledges and would like to express deep thanks to Robert Decker who supplied invaluable knowledge in regards to Kestrel’s usage for scramjet flows and combustion modeling methods.
This material is based on research sponsored by the USAFA and Virginia Polytechnic Institute and State University under agreement number FA7000-20-2-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Distribution Statement A. Distribution unlimited.

“The opinions, findings, views, conclusions or recommendations contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of USAFA or the U.S. Government.” (FA7000-20-2-0013)
Contents

List of Figures xiii

List of Tables xviii

1 Motivation and Material Review 1
  1.1 Brief History of Hypersonics ............................ 1
  1.2 Actuation in Engines ..................................... 5
    1.2.1 Variable Inlets ..................................... 8
    1.2.2 Variable Nozzles ................................... 9
  1.3 Combustors ............................................. 10
  1.4 Focus of Motivation .................................... 13

2 Hypersonic Modeling Basal 15
  2.1 CReSS ................................................ 15
  2.2 RANS CFD ............................................ 16
    2.2.1 Governing Equations ................................ 18
    2.2.2 Menter’s Shear Stress Transport .................... 20
    2.2.3 K-Omega .......................................... 20
    2.2.4 K-Epsilon ....................................... 21
2.3 CFD Solvers  ........................................................................ 22
  2.3.1 HyFOAM  ........................................................................ 23
  2.3.2 Kestrel  ........................................................................ 23

2.4 Mesh Generation Software  ................................................... 24
  2.4.1 SnappyHexMesh  .............................................................. 24
  2.4.2 Gmsh  ........................................................................... 25
  2.4.3 Pointwise  ....................................................................... 26
  2.4.4 Mesh Review  ................................................................. 27

3 CFD Solver Validation  .............................................................. 29
  3.1 Test Case 1  ........................................................................ 29
    3.1.1 Case Setup  ................................................................. 30
    3.1.2 Case Results  .............................................................. 31
  3.2 Test Case 2  ........................................................................ 35
    3.2.1 Case Setup  ................................................................. 36
    3.2.2 CFD Solver Limitations  ............................................... 37
    3.2.3 Case Results  .............................................................. 39

4 Testing Methodology  ................................................................. 40
  4.1 Engine Geometry  ............................................................... 41
  4.2 Required Inputs  ............................................................... 42
B.1.2  Tap-Reader.py ................................................. 72
B.1.3  Grid-Maker.py ................................................. 74
B.1.4  Hyslop-Comparison.py ........................................ 77

B.2  CRess Morphing Combustor Code Repository .......................... 80
B.2.1  Morphing-Combustor.py ......................................... 81
B.2.2  fileGen.py .......................................................... 84
B.2.3  MandP-MM.py ....................................................... 86
B.2.4  MM-Aggregate.py ................................................. 90
B.2.5  MM-csvread.py ................................................... 95
B.2.6  MM-Data-Analysis.py .......................................... 97
B.2.7  Morph-Contour.py ............................................... 99
B.2.8  Geom-Analysis.py .............................................. 102
List of Figures

1.1 The X-15A-2 platform was used for early ramjet investigations. a) X-15A-2 in flight[1], b) Ramjet mounting pylon damage post flight[9]. .......................... 2

1.2 The J58 used external piping, shown, to redirect pressurized flow to the combustor volume. [13] .......................... 3

1.3 The variable veins in the compressor portion of the B-58’s four engines allows for reactive performance a) The B-58 ’Greased Lightning’ at the SAC air museum, b) Armatures of the variable veins in the compressor housing. Pictures taken 08/09/2017. .......................... 5


1.5 Flow diagram for TBCC engine with out of flow turbine placement from proposed SR-72 drone. [34] .......................... 7

1.6 Shape-morphing structure with SiC–SiC face sheet, superalloy support structure, and shape memory actuating struts. [27] .......................... 8

1.7 External Compression Inlet. [27] .......................... 9

1.8 Morphing Stages of the SR-71 J58 Pratt and Whitney Engine. [26] .......................... 10

1.9 Contour plot of Static Temperature for different L/D ratios L/D=3, L/D=5 L/D=7 and L/D=9 with k-ε turbulence model [22]. .......................... 11

1.10 Contour plot of Mach number for M=2, M=2.25, M=2.5, M=2.75 and M=3 [22]. .......................... 12

xiii
1.11 Flow through analysis of HIFiRE combustor volume geometry exemplifying
an angle backsplash in a flame holder cavity.

1.12 Reacting flow through a combustor with boxed cavity geometry. [7]

2.1 Series of reactors used to model the system. [10]

2.2 Different modeling results from RANS, LES, and DNS solutions. [3]

2.3 Errors generated with using the OpenFOAM program SnappyHexMesh with
the forward fact of the cavity a. and the back splash of the cavity b.

2.4 Gmsh generated meshes for Test Case 2 showing the total geometry a. and a
focused view of combustor volume b.

2.5 HiFiRE geometry with injectors designed in Autodesk Inventor a. and the
same geometry meshed in Pointwise b.

2.6 Visualized transition from Structured to Unstructured mesh near the wall of
a scramjet.

3.1 Not to scale graphic overview of ANU’s scalable scramjet test engine. [17]

3.2 Overlaid results from 78 mm Injector and 0 mm Post Injector Length config-
uration.

3.3 Mach, Temperature, and Velocity CFD results for 78 mm Injector and 0 mm
Post Injector Length configuration.

3.4 Overlaid results from 78 mm Injector and 50 mm Post Injector Length con-
figuration.
3.5 Mach, Temperature, and Velocity CFD results for 78 mm Injector and 50 mm Post Injector Length configuration. .................................................. 33

3.6 Overlaid results from 48 mm Injector and 117 mm Post Injector Length configuration. .................................................. 34

3.7 Mach, Temperature, and Velocity CFD results for 48 mm Injector and 117 mm Post Injector Length configuration. .................................................. 34

3.8 Descriptive drawing of the tested geometry. [14] ................................................................. 35

3.9 Wireframe drawing of the tested geometry showing the Step (untested) and Divergent (tested) models. [14] .................................................. 36

3.10 Inlet a., Wall b., and Outflow c. surface declarations for tare case modeling. ................................................................. 37

3.11 Extreme pressure values shown to be forming within isolator section of scramjet without a. and with b. mesh shown. .................................................. 38

3.12 Overlaid values of mass flux scaled tare case results from the experimental data to that of the generated data from Kestrel. [14] ................................................................. 39

4.1 Variables in Geometry Selection; L is Length of cavity upper, H is the Depth of the cavity referencing off of the throat to the cavity, and R is the Angle of the ramped rear wall. ................................................................. 40

4.2 Simplified diagram of the combustor at different points of cavity volume with (a) being nearly fully expanded, (b) being at midpoint of travel, and (c) nearing the end of morphing range. [31] ................................................................. 41

4.3 Cross-sections of the HiFiRE geometry (a) and the different stages of the morphing combustor: 0.025m (b), 0.065m (c), and 0.11m (d). [31] ................................................................. 42
4.4 Correlation of input efficiency data [30] (a) and the standard day geometric altitude versus flight Mach number contours for constant freestream mass flow per unit area. A rough approximation of the range of Altitudes to Machs tested is marked by the red rectangle. [35] (b) .......................... 43

4.5 P2 Pressure: Altitude Versus Inlet Mach .......................... 44

5.1 Flow stations in a Scramjet engine. [30] .......................... 46

5.2 Selected Momentum Balance Versus Altitude charts (higher is better) for Mach 4.5 (a) Mach 6 (b) Mach 7.5 (c) Mach 9 (d) .......................... 47

5.3 Selected Temperature Versus Altitude charts (lower is better) for Mach 4.5 (a) Mach 6 (b) Mach 7.5 (c) Mach 9 (d) .......................... 48

5.4 Thrust Performance Maps: Stage 2 to Stage 4 without (a) and with HiFiRE (b), Stage 2 to Stage 9 without (c) and with HiFiRE (d). Decoder (e). .......................... 49

5.5 CFD results from combustion case simulation of the HiFiRE axis-symmetric alternative scramjet geometry showing a. Mach, b. Pressure, and c. Temperature. .......................... 50

5.6 Mach values from CFD Results at external Mach of 6 and Altitude of 80,000 ft: HiFiRE (a), Small (b), Medium (c), and Long Morphing Combustor (d). .......................... 52

A.1 Momentum Balance Versus Altitude by Mach (a) Mach 2.5 (b) Mach3 (c) Mach 3.5 (d) Mach 4 (e) Mach 4.5 (f) Mach 5 (g) Mach 5.5 (h) Mach 6 (i) Mach 6.5 (j) Mach 7 (k) Mach 7.5 (l) Mach 8 (m) Mach 8.5 (n) Mach 9 (o) Mach 9.5 (p) Mach 10 (r) Mach 10.5 (s) Mach 11 (t) Mach 11.5 .......................... 65
A.2 Combustor Chamber Temperature Versus Altitude by Mach (a) Mach 2.5 (b) Mach 3 (c) Mach 3.5 (d) Mach 4 (e) Mach 4.5 (f) Mach 5 (g) Mach 5.5 (h) Mach 6 (i) Mach 6.5 (j) Mach 7 (k) Mach 7.5 (l) Mach 8 (m) Mach 8.5 (n) Mach 9 (o) Mach 9.5 (p) Mach 10 (r) Mach 10.5 (s) Mach 11 (t) Mach 11.5.

A.3 CFD results from tare case simulation of tested scramjet geometry showing a. Mach, b. Pressure, and c. Temperature.
List of Tables

2.1 HyFOAM Flow Solver Details [15] .......................... 23
2.2 Kestrel Flow Solver Details [5] ............................. 24
3.1 List of tested geometries from Test Case 1 with those highlighted being replicated for validation. [17] .......................... 31
3.2 Combustor area ratios for Step and Divergent scramjet geometries [14]. . . . 35
3.3 Combustor operating conditions for experimental runs [14]. . . . . . . . 36
5.1 Momentum Balance (MB) and Temperature (T) values from CFD Results . 51
List of Abbreviations

\( \delta_{ij} \) Kroneker Delta

\( \epsilon \) Rate of Dissipation of Turbulence Kinetic Energy

\( \epsilon_c \) Critical Strain

\( \mathcal{P} \) Production Term

\( \mu \) Total Viscosity

\( \mu_l \) Laminar Viscosity

\( \mu_t \) Turbulent / Eddy Viscosity

\( \omega \) Turbulence Vorticity

\( \rho \) Density

\( \rho_c \) Critical Radius of Curvature

\( \tau_{ij} \) Reynolds Stress

\( \tau_{ik} \) Turbulence Shear Stress

ANU Australian National University

CREATE – AV Computational Research and Engineering Acquisition Tools and Environments for Air Vehicles

DMRJ Dual-Mode Ramjet Combustors

DNS Direct Numerical Simulation
DoD  Department of Defence

$H$  Total Enthalpy

$K$  Turbulence Kinetic Energy

KCFD  Kestrel CFD solver

LES  Large-Eddy Simulations

$p$  Pressure

$PC$  Power Curve

$Pr$  Prandtl Number

RANS  Reynolds-Averaged Navier-Stokes

$S_x$  Source Term

$S_{ij}$  Strain Rate Tensor

SST  Menter’s Shear Stress Transport

$T$  Temperature

$t$  Thickness of Ceramic Weave Sheet

TKE  Turbulence Kinetic Energy

$u_i$  Velocity Components

$y+$  Modeling constraint for boundary layer modeling

DARPA  Defense Advanced Research Projects Agency

HTV  Hypersonic Technology Vehicle
Chapter 1

Motivation and Material Review

Losing the restriction in performance capacity is a common theme when it comes to looking into variable solution motivations. The goal of making an engine operate well out of the range of its brethren whether it be in efficiency or raw speed is highly enticing. Especially if it can be achieved through simplistic means. From variable internal flows, inlets, nozzles, and even combustors, a wide gambit of variable geometry solutions have been discussed though few have ever been implemented in real world operation. Historically modification of scramjet or ramjet functions internally has not occurred intentionally when focusing upon single state engines. Be this for a variety of reasons such as weight and/or material survivability as seen in the X-15 tests [9]. The motivation for this paper is one for pure analysis of the potential that a variable geometry solution could garner and to what results could be expected.

The following is a deep dive into the history of ram and scramjet implementations followed by analysis of variable geometry implementations with a discussion of the apparent and unfortunate lack of publications in this subject matter.

1.1 Brief History of Hypersonics

The expansion into the realm of hypersonic propulsion for air breathing crafts has seen multiple upstarts and stumbles through the past century. A major component of this timeline is the X-15A-2 project which used a modified X-15 platform to house a ramjet for testing,
seen in Fig 1.1. In this flight with the ramjet activated the aircraft achieved Mach 6.72 but not without severe loss. The shell of the plane was scorched along its length and the ramjet had dramatically and destructively detached itself [9]. This had occurred due to materials which where discovered to be grossly inadequate for the temperatures experienced. High temperatures led by shock/shock interactions which where experienced and categorized for the first time. Though this is certainly not the first experiment into ramjets, scramjets, and hypersonic flight the stage was set and an understanding of the future problems faced were beginning to be actualized.

An engine which has a notable part in the history of variable geometry engines is the Pratt and Whitney J58. Notably used in the SR-71; this engine accelerated the airframe up to Mach 3.2 from take off. Accomplishing this task was through the clever usage of redirection of flow combined with novel usage of combustor volume choice [26]. Taking from the third stage of the compressor the pressurized flow was directed to a volume post the turbines combustor so to act like a ramjet combustor volume. These redirecting tubes are shown in Fig 1.2. This was not the only moving portion of the engine as this was combined with a variable inlet which actuated its cone so that the shock correctly flowed into the engine for
smooth operation at upper Mach speeds[26]. Further, a variable nozzle was also implemented so that the combustor operated at its peak performance throughout its operating stages. All of this combined to an engine which was in operation for 25+ years and stands out still as one of the few ever fully implemented morphing geometry engines. For this case, variable vein interactions, seen implemented on the B-58 in Fig 1.3, are not seen as variable geometry to the level of this projects restrictions as they do not radically modify the flow through the engine in a manner which can see massive state change such as from turbine to ramjet.

Expanding into the more modern approaches to hypersonic testing is the FALCON Hypersonic Technology Vehicles (HTV’s) project which saw a step into the modern era. This project was the repeated testing of prototype airframe test beds for the purpose of furthering hypersonic understanding mainly with a focus for control surfaces[33][11]. Initially the project was focused on glider control surface interaction with the HTV-1 project. Finding the initial testing capable, a redesigned HTV-2 airframe was developed but was once again unpowered. Finally with the introduction of the HTV-3 redesign and an adjustment in Defense Advanced Research Projects Agency (DARPA) focus, the project became the test bed
for scramjets and other hypersonic propulsion systems[33]. Using a Turbine Based Combine Cycle (TBCC) power plant the airframe was capable of taking off from a runway and switching in flight to its scramjet engine. This was achieved by having the turbine positioned in such a manner that at high Mach speeds the flow path was directed in a manner to exclude the turbine. This allows for the scramjet to operate without flow complications that are seen with turbines positioned in the way for the flow path as seen in the J58 engine. The operational speed of the airframe was Mach 6 once acting as a pure scramjet[33]. Interestingly the engine was designed to function as a dual-mode ram/scramjet so to allow for it stably operate at lower Mach speeds of around Mach 2. Ultimately, the project was ended after repeated in flight losses of control resulting in being rolled into the ocean and destroyed. A final flight test was canceled and a proposed redesign was never produced due to mounting costs and a belief that nothing more would be further learned from repeated tests[33].

A notable project to discuss in the brief history of hypersonics is the HIFiRE Flight 2 project. This project achieved Mach 8 scramjet operations while concurrently providing a high-quality data set of a scramjet operating through a dual-to-scram mode transition[18]. The data set from this project has gone on to be referenced and modeled in a variety of different manners. From Large-Eddy Simulations (LES) simulations modeling full combustion and flow[4] to 1D solvers such as CReSS[10] using the data as validation of modeling accuracy. The dataset is one of the few ever created due to the extreme cost required for hypersonic testing, ground or in flight[9]. The innate risk and cost has led to most in the field seeking towards different modeling techniques to simulate hypersonic flows. Unfortunately, this opens one up to a higher risk when it comes to real world testing based on these models[9].
1.2. Actuation in Engines

The allure of making an any situation hypersonic capable engine is certainly not lost on the world. The most difficult to achieve is an engine which can take off and then accelerate to a high enough Mach speed to initialize the ramjet engine which then can transition into a scramjet engine. The two trains of thought when it comes to the implementation is to either have the turbine engine used for take off encased in the flow, similar to that seen with the J58, or to have the turbine placed into its own flow path as used in the FALCON project. Chen et al proposed a design in a manner in line with the former as shown in Fig 1.4. The main drawback of this placement is a dramatic increase in complication of actuation and flow modulation controls. The biggest benefit as such is a far diminished volume requirement for the engine as it is all contained inline more or less. In the case of Chen et al the proposed engine relied on 11 variable geometries through out the engine, from inlet to nozzle[6]. In doing so the the engine was conceptually capable of operating from take off to Mach 5. Unfortunately many issues arise due to the placement of the turbine in the flow. Volume and flow restrictions are of main concern. Volume in that the requirement that the scram/ram jets flow around the encased turbine forces the size of the over all engines diameter to be dramatically accentuated. This is not taking into account
the required managing systems that must be located on the turbine and protected from the properties of the supersonic flow. Restrictions on flow also come to be a point of contention as due to the turbines positioning there is no possibility for smooth flow from isolator to the combustor of the ram/scram jet[6].

![Flow diagram for TBCC engine with inflow turbine placement.](image)

The counter to the inflow TBCC design is by having the turbine out of the flow of the scram/ramjet as seen in the proposed SR-72 hypersonic drone replacement engine in Fig 1.5. This drone has a shared inlet and outlet but otherwise the two engines, turbine and dual-mode jet, operate entirely without interaction allowing for clear flow for the ram/scramjet from inlet to nozzle. This is achieved all the while reducing complexity of operation. With the only detriment of this style of design being overall volume requirements. The trade off between the in and out of flow TBCC in regards to weight seems negligible with the removal of all support systems that are required for the multitude of variable geometry actions for the in flow turbine counterbalancing the volumetric difference of the out of flow turbine.

Stepping away from TBCC based scramjet engines to focusing purely on ramjets, scramjets, or dual-mode engines, different technologies come to light which allow for the variation of these engines so to match a variety of conditions and react accordingly. One of these such technologies is the ability to smoothly vary the wall structure of a scramjet through the usage of mesh like ceramic composites[27]. This meshing structure relies on smoothly pinned jointing interlaced ceramic composites in the weave structure. The jointing allows for
1.2. Actuation in Engines

![Flow diagram for TBCC engine with out of flow turbine placement from proposed SR-72 drone.][1]

Figure 1.5: Flow diagram for TBCC engine with out of flow turbine placement from proposed SR-72 drone. [34]

...the testing of the superalloy strut structures with a Mach 8 flow at 800K which resulted in a highly variable geometry with reliable control and little to no deflection. Equation 1.1 is the formula to determine the deflections capable where $\rho_c$ is the critical radius of curvature, $t$ is the thickness of the sheet, and $\epsilon_c$ is the critical strain. Miles et al limited the critical strain to % 0.25 as it was less than half that of the tested composites[27]. Composited ceramics being managed in this manner allow for smooth modifications of the flow through a scramjet with the conforming mesh shown in Fig 1.6. Though only tested for isolator like situations the ability to concisely constrain flow characteristics in high pressure and high temperature settings lends towards internal modifications to better react to situational requirements.

$$\rho_c = \frac{t}{2\epsilon_c}$$ (1.1)
1.2.1 Variable Inlets

A main focus of many who wish to make variable components for scramjet operations is inlet configuration. Whether it be a waverider flow interactions to more innocent shock capture, the lack of extremes in regards to both pressure and temperature gradients allows for a wide gambit of solutions to be proposed. With waverider streamline flow interactions, which rely heavily on the rate and manner of capture, the lift capabilities of the airframe become jointly reliant. Through the use of external compression inlets which do not capture all of the incoming streamlines as seen in Fig 1.7, waverider style scramjet platforms can use the uncaptured streamline flows to lift the vehicle as they pass below the airframe. Variable geometry solutions seek to find a manner by which to reach higher Mach values with out sacrificing the fine lift capability control. Their static brethren must operate within a restrictive range to support the lift requirements of the respective airframe. Research into this has shown that with increasing Mach values, a decreasing inlet shock angle was needed to correctly capture the appropriate amount of streamline flow[24]. This was due to the streamline flow, with its respective pressure and speed, having a far greater affect on the
1.2. ACTUATION IN ENGINES

Figure 1.7: External Compression Inlet. [27]

Lift of the craft as the Mach speed increases. Reducing the volume which ‘escapes’ the inlet lowers the amount that the waverider can use for lift but due to the higher speed experienced proportionally its equivalent to lower speeds interactions with high angle inlet geometries. Doing so dramatically increases the operational capabilities of aircraft without over complicating the design.

1.2.2 Variable Nozzles

A similar line of reasoning has been applied to variable nozzle designs. The SR-71’s J58 has a variable inlet and nozzle as the increase in efficiency cannot be ignored. As shown in Fig 1.8 the nozzle and inlet dramatically change angle and position respectively. Focusing on the nozzle interactions, in 1D testing for hypersonic scramjets it was found that should any internal geometrical modulation occurring to the flow’s path due geometry variation or situational disturbances results in the nozzle possibly dramatically hindering the performance of the engine[31]. Once more falling into the very glaring pattern that variable geometry is the answer to make effective and adaptable solutions, nozzle modulation must occur to make successful long term solutions. Unfortunately, research in hypersonic variable nozzle design is not well defined. Due to the variety of nozzle designs and overall lack of real world testing, incentive is lacking to expand into a difficult to physically test design. Optimizations
in scramjet nozzle design have occurred but in the focus matching to the combustor so a theoretical performance value can occur and be optimized[29].

![Figure 1.8: Morphing Stages of the SR-71 J58 Pratt and Whitney Engine. [26]](image)

### 1.3 Combustors

The largest paucity in available data in regards to variable geometry for hypersonic jets is the combustor. Its high heat, pressure, and turbulence makes it a highly difficult member to accurately simulate. Due to this, multiple models treat it as a black box when it comes to CFD simulations[24]. This method leaves much to be desired as it entirely assumes
Figure 1.9: Contour plot of Static Temperature for different L/D ratios L/D=3, L/D=5, L/D=7 and L/D=9 with k-ε turbulence model [22].

Combustor reactions and flow. Similar to that of the nozzle design standards mentioned previously, combustor volume design is usually limited to a restrictive range of operation[29]. The cases which attempted CFD simulation of the combustor and combustion itself where found to be mainly limited to RANS k-ω modeled simulations and without true injection modeling[16][19][29][7]. Although the reigning standard, other simulations have occurred which cover the whole engine and support combustion[28][20]. One such project, relied on LES simulation to model the HIFiRE engine operating from Mach 6.5 to Mach 8. This was combined with the FPVA compressible combustion model to handle the methane ethylene chemical reaction. This was unfortunately to be an inaccurate combustion model. Though when forced to assume near instantaneous combustion the LES simulation was found to be far more agreeable with the published HIFiRE data[4]. Another simulation option, though not CFD, is the 1D Solver CReSS[10]. With this solver 1D morphing combustor volumes
have been explored focusing on flame holder cavity volume interactions. It was found that dramatic variance in performance existed and was more or less untapped potential\[31\].

A major contention in combustor design and modeling is the handling of flame holder cavities. The point of conflict in the manner is how the wall towards the back of the cavity, being the back is seen as the wall furthest from the inlet, is configured. Two major schools of thought exist with either an angled backsplash\[23\], as seen in Fig 1.11, or a perpendicular backsplash\[7\], as seen in Fig 1.12. Most literature and real world testing agrees that an angled backsplash performs the best as it allows for the flow to easily evacuate the volume in
1.4 Focus of Motivation

Hypersonic data is in itself sparse due to its many issues and costs associated with the gathering of real world testing results. Developments on the scraps of released data has a smooth and non-turbulent manner. Through testing it repeatedly showed that at higher Mach speeds, around 6 plus, the cavity began to become a hindrance for the stability of the flow resulting in higher held temperatures and pressures which negatively effected the performance of the engine[7]. 1D testing of variable geometry cavities saw much the same with the biggest factor being how the combustor was holding onto the heat in the flow[31]. The ability to shrink and expand the cavity’s volume dependent on flow conditions would alleviate the issues caused by the flame holder but keep its benefit of preventing unstarts at lower Mach speeds. Figures 1.10 and 1.9 show examples of planar-symmetric CFD results where in which it becomes extraordinarily visible how much the external velocity and the combustor’s cavity size effects the flow and combustion through the engine. In the previously mentioned case the temperature values for the different L/D ratios; results varied wildly with the coldest running case being L/D = 9 at a maximum of 1600 K [22]. In stark comparison, L/D = 3 and L/D = 7 both experienced peaks of around 3250 K. This is for the same inlet and injector conditions [22]. Just varying the inlet Mach dramatically changed the effective interference experienced by the combustor cavity, as shown in figure 1.10. Both of these experienced reactions lead towards credence in the morphing combustor concept.

Figure 1.11: Flow through analysis of HIFiRE combustor volume geometry exemplifying an angle backsplash in a flame holder cavity.
lead to numerous non-variable geometry designs for very specific operational realms. This itself leading to repetitive designs as the cost to differentiate from a known solution has a very risk factor. When it comes to variable geometry designs the line share are focused on inlet interactions with a special focus towards waverider operation as that is currently a heavily researched style of hypersonic airframe. When it comes to the hot side of the engine there is a stark lack of resources both in raw data and in publications mainly due to the extreme difficulties of accurately modeling the flow and combustion. Of what is published there is little focus on combustor volume variation, with exception to Mahto et al. [22], and instead a focus on nozzle modification. This apparent gap in knowledge and testing for combustor flow interactions leaves a figurative open door for investigation where the focus is on simple movement based modifications of a scramjet combustor cavity. What this proposes is observations of a part which in combination with the studies of scramjet intake and nozzle designs can be combined into a more adaptive and capable engine.
Chapter 2

Hypersonic Modeling Basal

For hypersonic flows an understanding of the modeling conventions and expectations must be understood. This chapter seeks to explain the make up of the modeling techniques and formulations used to predict the interactions seen in hypersonic flows that are later tested in section 3. From CReSS to CFD a multitude of different solutions are proctored to solve the worlds pressing hypersonic conundrums. The basis for the CFD work is done with solvers using the RANS modeling technique, extrapolated in section 2.2. For this the background formulation must be understood in regards to the implementation of the different solvers. This also applies to the Menter’s Shear Stress Transport method used. Discussion in regards to the progression of workflow development of mesh generation and refinement will occur later on in this chapter as well to fully explain the roadblocks experienced and their respective solutions.

2.1 CReSS

The simulation of the multiple Scramjet geometries was performed through the usage of the quasi-1D Canonical Reactor Scramjet Simulation (CReSS) solver [10] [31]. CReSS uses a reaction rate limited combustion model through a series of reactors as seen in Fig: 2.1 to simulate the non-equilibrium behavior present in high speed combustion. An important note on the plug flow reactors is that they are formulated to include viscous effects and
for friction, in stark contrast with standard plug reactors. It most importantly supports the modeling of cavity based flame holders; a feature which most contemporary 1D solvers cannot perform\cite{10}. Due to this ability and its ease by which it can evaluate a variety of geometries, it fit the requirements of the project and delivered a fairly high resolution data set. Another boon is that CReSS’ required inputs are near identical to that of a standard CFD solver, if simplified, which allows for rapid transition from the development of CReSS cases to the respective CFD case counterpart. The inputs are described in detail in section 4.2.

![Figure 2.1: Series of reactors used to model the system.][10]

### 2.2 RANS CFD

The underlying method of CFD simulation for all of the tested CFD solvers is the Reynolds-Averaged Navier-Stokes (RANS) equations. This series of equations are a time averaged solution to model turbulent flows in fluid simulations. This method has rapidly become the standard by which dual-mode ramjet combustors (DMRJ) are designed and modeled \cite{22} \cite{5}. Reasons for this choice is the solution’s innate capabilities to handle extreme simulation requirements that are standard to that of DMRJ’s. The main theory by which RANS simulations function is that instead of simulating all scales of of the flow, referred to as Direct Numerical Simulation (DNS), or only the largest scales, seen with LES; instead the
RANS solution only models and resolves all of the turbulence and mean flow structures [5]. A visualization of the difference of results one can garner from these different solutions can be seen in figure 2.2. One the main benefits of this is the resulting drastic reduction in computational power requirements. LES has gained traction as computing power has proceeded to become more economically viable but DNS remains permanently unfeasible for even mildly complicated or medium scale simulations become computationally impossible.

![Different modeling results from RANS, LES, and DNS solutions.](image)

Figure 2.2: Different modeling results from RANS, LES, and DNS solutions. [3]

The unfortunate side effect of RANS simulations is an intrinsic sensitivity to a majority of model parameters. These being exemplified in those features such as the Schmidt number, chemistry modeling, and mesh quality [5]. To elaborate the Schmidt number controls the turbulent mixing rate, the chemistry modeling determines combustion and species mixing capabilities, and the mesh quality has a major deciding factors in regards to near wall interactions and internal flow characteristics. Mesh quality will be discussed in further detail later on in section 2.4. Due to these sensitivities it is dire that one must validate the results of their selected CFD solver. The best method of validation is through the usage of real world test data as comparing to others CFD solutions can result in a skewed perception
of supposed reality. Two examples of effectively testing and obtaining comparable to real world results can be seen in sections 3.1 and 3.2.

2.2.1 Governing Equations

The RANS solution has a series of governing equations which are expressed in the following formulae which are shown derived for compressible flow modeling. This is due to hypersonic simulations requiring a compressible flow model as a result of the experienced extreme pressures, temperatures, and velocities [22]. The continuity equation, Eq: 2.1, is what determines the transfer of turbulence properties [22] [12]. The momentum equation, Eq: 2.2, and the energy equation, Eq: 2.3, shown also describe the propagation of their respective feature sets in the simulated flow.

\[
\frac{\delta \rho}{\delta t} + \frac{\delta}{\delta x_k}(\rho u_k) = 0, k = 1, 2, 3
\]  
(2.1)

\[
\frac{\delta}{\delta t}(\rho u_i) + \frac{\delta}{\delta x_k}(\rho u_i u_k) + \frac{\delta P}{\delta x_i} = \frac{\delta \tau_{jk}}{\delta x_k}, i, k = 1, 2, 3
\]  
(2.2)

\[
\frac{\delta}{\delta t}(\rho H) + \frac{\delta}{\delta x_k}(\rho u_k H) = -\frac{\delta}{\delta x_k}(u_j \tau_{jk}) + \frac{\delta q_k}{\delta x_k}, j, k = 1, 2, 3
\]  
(2.3)

The viscosity or motion of the fluids being modeled have three governing equations. The first, shown in Eq: 2.4, is the simplest with total viscosity being made of the simple combination of the laminar and turbulent viscosity. Following this is Laminar viscosity which is the characterization of a smooth flow and is found near walls due to the lower velocity caused by the boundary layer interactions. This viscosity can be seen in Eq: 2.5. The final viscosity to
discuss is the turbulent or eddy viscosity. This viscosity, seen in Eq: 2.6, can be interpreted as the combination of turbulent eddies into a facsimile of a larger scale motion from which an effective turbulence viscosity is discerned\cite{22}. The eddy viscosity is the basis of an LES simulation where the focus of resolution is the turbulent viscosity. The near wall laminar viscosity would be modeled in the case of an LES simulation. RANS simulations effectively model all of the different viscosities\cite{5}.

\[
\mu = \mu_l + \mu_t \tag{2.4}
\]

\[
\mu_l = \mu_{ref}(\frac{T}{\delta T_{ref}})^{3/2}(\frac{T_{ref} + S}{T + S}) \tag{2.5}
\]

\[
\mu_t = c_n \frac{\rho K^2}{\epsilon} \tag{2.6}
\]

To complete the governing equations used by the RANS model, the turbulent shear stress, species transport equation, and source term definition are shown in equations 2.7, 2.8, and 2.9 respectively. With these governing equations, the RANS model can effectively model an entire compressible, turbulent, hypersonic flow \cite{5}.

\[
\tau_{ik} = \mu_t \left( \frac{\delta u_i}{\delta x_k} + \frac{\delta u_k}{\delta x_i} \right) \tag{2.7}
\]

\[
\frac{\delta}{\delta t}(\rho Y_n) + \frac{\delta}{\delta x_k}(\rho Y_n u_k) = \frac{\delta}{\delta x_k}\left[ (\frac{\mu_l}{P r} + \frac{\mu_t}{\sigma_c}) \frac{\delta Y_n}{\delta x_k} \right], k = 1, 2, 3 \tag{2.8}
\]
2.2.2 Menter’s Shear Stress Transport

Along side the usage of RANS based simulations being popular, the usage of Menter’s Shear Stress Transport (SST) turbulence model has also become, for lack of a better term, a household standard in regards to its usage in CFD solutions. The reasoning for this is its innate ability to easily, read stably, model low Reynolds number flows so to effective model the boundary layer interactions through the usage of K-ω model while still retaining the ability to adequately model the higher Reynolds number freestream flow [25]. This is one of the reasons the SST module is usually written as the SST K-ω turbulence model. The K-ω model is explained in further detail in section 2.2.3. That being said, The SST model does not rely solely on the K-ω turbulence as it uses the K-ε turbulence model, discussed in section 2.2.4, to model the free-stream as it is not as sensitive to inlet turbulence resolving as the K-ω model finds itself [25]. By combining these two two-equation turbulence models the resulting partnership garners a highly capable turbulence transport model which it’s robustness matches it ability to closely replicate real world results. The implementations of the SST model varies depending on CFD solver where it can be exemplified with how HyFOAM requires declared specific wall formulations to help manage the transition point between the two turbulence models and Kestrel does not [15].

2.2.3 K-Omega

The K-ω two equation turbulence model is an effective near wall and free stream model which is capable of handling low Reynolds number simulations but does retain the ability
2.2. RANS CFD

to model freestream flow [22][5]. Though it is able to model free stream flow, its sensitivity to source terms, such as inlets, makes it more sensitive and therefore difficult to resolve complex simulations than using an SST model. Seen below is its respective Turbulence Kinetic Energy (TKE) equation, Eq: 2.10, and turbulent vorticicty, ie omega, formulation seen in Eq: 2.15. Further, a product term, Eq: 2.12, is required and which relies on the Boussinesq Approximation seen in Eq: 2.13.

\[
\frac{\delta}{\delta t}(\rho K) + \frac{\delta}{\delta x_j}(\rho u_j K) = P - \beta^* \rho \omega K + \frac{\delta}{\delta x_j}[(\mu + \sigma_K \mu_t) \frac{\delta K}{\delta x_j}]
\] (2.10)

\[
\frac{\delta}{\delta t}(\rho \omega) + \frac{\delta}{\delta x_j}(\rho u_j \omega) = \frac{\gamma}{\nu_t} P - \beta^* \rho \omega^2 + \frac{\delta}{\delta x_j}[(\mu + \sigma_\omega \mu_t) \frac{\delta \omega}{\delta x_j}] + 2(1 - F_1) \frac{\rho \sigma_\omega^2 \delta K}{\omega} \frac{\delta \omega}{\delta x_j} \frac{\delta \omega}{\delta x_j}
\] (2.11)

\[
P = \tau_{ij} \frac{\delta u_i}{\delta x_j}
\] (2.12)

\[
\tau_{ij} = 2\mu_t(S_{ij} - \frac{1}{3} \frac{\delta u_K}{\delta x_K} \delta_{ij}) - \frac{2}{3} \rho K \delta_{ij}
\] (2.13)

2.2.4 K-Epsilon

The K-\(\epsilon\) two equation turbulence model is a highly effective free stream model which is easily capable of handling high Reynolds number simulations [22][5]. It retains an innate lack of sensitivity to inlet flow causing discrepancies though it struggles to effectively resolve near wall and low Reynolds number scenarios. Seen below is its respective TKE equation, Eq: 2.14, and rate of dissipation of TKE, ie epsilon, seen in Eq: 2.15. The difference between
the K-\(\epsilon\) and K-\(\omega\) models can be seen to be slight but the effective use cases vary heavily. Though, with that stated, the K-\(\omega\) model can be used solely for an entire simulation where it effectively models the boundary layer and the K-\(\epsilon\) cannot.

\[
\frac{\delta}{\delta t}(\rho K) + \frac{\delta}{\delta x_k}(\rho u_k K) = \frac{\delta}{\delta x_k}[(\frac{\mu_l}{Pr} + \frac{\mu_t}{\sigma_K}) \frac{\delta K}{\delta x_k}] + S_K \tag{2.14}
\]

\[
\frac{\delta}{\delta t}(\rho \epsilon) + \frac{\delta}{\delta x_k}(\rho u_k \epsilon) = \frac{\delta}{\delta x_k}[(\frac{\mu_l}{Pr} + \frac{\mu_t}{\sigma_\epsilon}) \frac{\delta \epsilon}{\delta x_k}] + S_\epsilon \tag{2.15}
\]

2.3 CFD Solvers

When it comes to hypersonic CFD simulation there are a wide selection of solvers which are currently able to replicate, with acceptable accuracy, real world results. Examples can be seen in SU2, ANSYS Fluent, VULCAN, CFD++, STAR-CCM+, HyFOAM, and CREATE Kestrel to name a few. For this project only two CFD solvers came into focus for the project which will be discussed in detail in their respective subsections. HyFOAM, an extension of the OpenFOAM project is mainly focused upon modeling interactions of re-entry vehicles but has been shown to effectively model select internal scramjet flows [15]. CREATE Kestrel is the second tested solver which became a focus for the project and is highly adept at modeling internal scramjet flows with combustion modeling [5]. Both of these solutions have been tested by others and have shown to perform admirably in respective scenarios. With this statement, one must still verify and discover effective workflow solutions for these solvers. Later on in chapter 3 both solvers are shown being tested for functionality and accuracy.
2.3. CFD SOLVERS

2.3.1 HyFOAM

As mentioned before HyFOAM is an OpenFOAM based hypersonic CFD solver. HyFOAM uses Menter’s SST turbulence model, expanded upon in section 2.2.2, due to the model’s ability to handle a variety of unorthodox geometries that are commonly presented in scramjet design \[15\]. To make the SST turbulence model perform to its best ability flux limiters are applied to force convergence by smoothing edge case extreme values; these limiters being named in Table 2.1. Later discussed in detail, section 3.2.2, HyFOAM was found to not be capable to manage cavity based combustor volume scramjet examples adequately. As well, the combustion modeling capabilities are unfortunately lacking, mainly due to a severe lack of consistent documentation and the open-source nature of its parent software, OpenFOAM.

<table>
<thead>
<tr>
<th>Inviscid Flux Scheme</th>
<th>Kurganov and Tadmor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inviscid Flux Limiter</td>
<td>Gamma1 and van Leer</td>
</tr>
<tr>
<td>Implicit Time Integration</td>
<td>First-Order Accurate Euler</td>
</tr>
</tbody>
</table>

2.3.2 Kestrel

Kestrel is a CFD solver from the Computational Research and Engineering Acquisition Tools and Environments Air Vehicles (CREATE-AV) program which was started by the High Performance Computing Modernization Program of the Department of Defence (DoD)\[5\]. Its was built around the concept of being an event based architecture where each internal component was made separate. Due to this, Kestrel was not built as singular overall code suite but is instead a collection of individual subsections. This allows for the code suite to cover a variety of complex, multi-physics simulations with the ease of managing only smaller portions of code. The benefit to the user is the software is also designed from the start to be easy to use with a comprehensive graphic user interface. The individual limiting functions
for the Kestrel CFD solver (KCFD) are shown in table 2.2.

<table>
<thead>
<tr>
<th>Inviscid Flux Scheme</th>
<th>HLLE++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inviscid Flux Limiter</td>
<td>Barth/Jespersen with Custom Shock Switch</td>
</tr>
<tr>
<td>Implicit Time Integration</td>
<td>Point-Implicit with Subiterations</td>
</tr>
<tr>
<td>Time Step</td>
<td>Global CFL Number</td>
</tr>
<tr>
<td>Gradient Reconstruction</td>
<td>Inverse Distance Weighted Least-Squares</td>
</tr>
</tbody>
</table>

Table 2.2: Kestrel Flow Solver Details [5]

2.4 Mesh Generation Software

The development of the workflow for the generation of effective and consistent meshes has been found to be not a straightforward task. The limitations experienced with the tested mesh generation software will be discussed further in the following sections. Two different open source solutions were tried with SnappyHexMesh (a OpenFOAM based meshing solver) and Gmsh which is a 3D Finite Element Mesh Generator. Following insurmountable roadblocks with these previously mentioned pieces of software, Pointwise became the main mesh generation software for this project.

2.4.1 SnappyHexMesh

SnappyHexMesh is a piece of software from the OpenFOAM project which one can supply with the model of what they wish to mesh in the form of STL files. These files are then interpreted, and combined with the declaration of different facial planes, used to create a singular mesh. Working with SnappyHexMesh was found to be troublesome when generating a smooth mesh of a planer-symmetric design that had faces which were found to be not flat and/or well defined. An example of the issues experienced with SnappyHexMesh can be seen in figure 2.3. Various test cases and troubleshooting attempts were performed but
ultimately it was found that the meshes created were temperamental at best and overall unfit for simulating which forced a move to a different meshing software.

![Image](image-url)

Figure 2.3: Errors generated with using the OpenFOAM program SnappyHexMesh with the forward face of the cavity a. and the back splash of the cavity b.

### 2.4.2 Gmsh

This then led to an initial switch to Gmsh as it was capable of cleanly declaring a geometry, generating a mesh to fill said geometry, and then being able to convert to a OpenFOAM compatible version. Unfortunately this solution had two main issues; the first being that memory management was poor which resulted in some cases where hundreds of gigabytes of ram were required to mesh with decent density. This led to situations where over a terabyte of ram was found to be not enough. The second was an issue with the generation of prism layers, sometimes referred to as inflation layers or near wall cells; an example can be seen in figure 2.6. These prism layers are used to well establish the boundary layers of the mesh. The issue with Gmsh came into focus when meshing the combustor cavity volume of the tested geometry which contained both an obtuse and an acute sharp angle seen in figure 3.9. These sharp angles made it such that configuring Gmsh to generate said prism layers became incorrigible and ultimately made it such that well defined boundary layers
could not be established in a clean and smooth manner. This compounded with Gmsh’s memory mismanagement forced a move to a paid for and more industry standard solution. An example of a mesh made with Gmsh can be seen in figure 2.4 without a prism layer.

![Figure 2.4: Gmsh generated meshes for Test Case 2 showing the total geometry a. and a focused view of combustor volume b.](image)

**2.4.3 Pointwise**

Pointwise is a 3D mesh generation tool which is a licensed solution which supports a variety of CFD solver focused attributes. This makes it exceptional as it makes the generated meshes best match the CFD solver used. The workflow for Pointwise is similarly accessible. Using Autodesk Inventor a geometry can be easily generated to that of the researcher’s direct specifications. This is then exported as an IGES file in the wireframe format. IGES files can be directly imported into Pointwise which allows for the retention of critical geometry features such as injector size and location. The workflow is exemplified in the figure 2.5 where a before and after meshing model is shown. The meshing is then done through the usage of the T-Rex meshing algorithm which generates a prism layer on declared walls and then fills the rest of the volume with an effective unstructured mesh.
2.4. Mesh Generation Software

Figure 2.5: HiFiRE geometry with injectors designed in Autodesk Inventor a. and the same geometry meshed in Pointwise b.

2.4.4 Mesh Review

To clarify more on the type of meshes used in this project; the main body, all of the volume outside of the near wall volumes, is comprised of an unstructured mesh. An unstructured mesh is characterized by containing tetrahedral cells and said cells being formed in an arbitrary manner. In stark contrast is a structured mesh where the cells are hexahedral and are created in an implicit manner. These two methods of mesh generation serve different purposes. In the case of this project, to control boundary layer interactions it is required that a prism layer be established which must be made up of structured cell layers. On average, these layers are extended to around one centimeter from the wall of the scramjet. This is so a virtual Coanda effect does not occur where in which the shock train of the flow within the
scramjet body is not able to stabilize due to the boundary layer incorrectly modeling and not adhering and/or releasing the flow properly [8]. A further facet of the prism layer and its generation in regards to the RANS CFD simulations is that the $y+$ value needs to be near one [22]. The $y+$ value can be conceptualized as a modeling constraint for correct near wall cell spacing for congruous boundary layer modeling [21]. Due to the usage of the RANS CFD approach, it is necessary to completely model the viscous sublayer which is dominated by the linear velocity law. The viscous sublayer effects are seen in the range of $y+$ less than five [21]. In the testing described in section 5.2, the prism layer was modeled so to contain a $y+$ value of three or less. Outside of the near wall values the unstructured mesh fills the rest of the region. This is a fast and effective method to build detailed meshes with lowered memory requirements and cell count.
Chapter 3

CFD Solver Validation

When using any piece of software it is necessary to verify and validate one's results. For this project, real world scramjet test data without combustion was highly necessary as it is far quicker to begin with tare cases when learning new software suites and workflows. Two such cases where found to meet this criteria; with both being initial tested with HyFOAM. The first case is based on the paper by Peter Hyslop called CFD Modelling of Supersonic Combustion in a Scramjet Engine [17] and will be reviewed in section 3.1. The second is a paper by Mark Gruber, Stephan Smith, and Tarun Mathur called Experimental Characterization of Hydrocarbon-Fueled, Axisymmetric Scramjet Combustor Flowpaths [14] and will be reviewed in section 3.2. This paper in particular became a major focus of the project as it led the development of new meshing techniques, the movement to a new CFD solver, and the development of internal combustion modeling capabilities with Kestrel. Both of these cases proved deeply influential in regards to development of CFD workflows for case setup and review.

3.1 Test Case 1

This paper attempted to determine the effectiveness of 1998 CFD solutions in regards to the ability to effectively resolve hypersonic simulations with and without combustion modeling while also testing novel theories in scramjet injector positioning and nozzle thrust surface
angles. This was performed using a real world scramjet test cell, represented in figure 3.1, that could be reconfigured in a variety of different geometric positions to vary both the injector body length and the length post injector before that of the exhaust nozzle / thrust surface.

![Figure 3.1: Not to scale graphic overview of ANU’s scalable scramjet test engine.](image)

### 3.1.1 Case Setup

Three individual cases were replicated as they were the ones by which real world testing occurred for flow through, or tare case evaluation. These configurations are demarcated in Table 3.1. From this the initial conditions of the case were as such; inlet initial conditions where a pressure of 70 kPa, a temperature of 1217 K, and a velocity of 1697 m/s. The walls were simulated with a no slip condition and an isothermal wall temperature of 298 K. The air mixture was set at 21% Oxygen and 79% Nitrogen mixture [17]. The software used to model this test case was HyFOAM, described in section 2.3.1, as Kestrel had not yet become available to this project. Each case was ran for a simulated 1 ms as by that time the case
3.1. Test Case 1

Table 3.1: List of tested geometries from Test Case 1 with those highlighted being replicated for validation. [17]

<table>
<thead>
<tr>
<th>Injector Length [mm]</th>
<th>Flat Duct Length [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>78</td>
<td>50</td>
</tr>
<tr>
<td>78</td>
<td>87</td>
</tr>
<tr>
<td>48</td>
<td>80</td>
</tr>
<tr>
<td>48</td>
<td>117</td>
</tr>
</tbody>
</table>

had resolved to values from which a determination of acceptable response could be made.

3.1.2 Case Results

The main point of comparison in this project is comparing to that of the experimental non-combustion data-sets. The reasoning for this is that the time of testing this validation case, combustion modeling had not been fully configured and understood for use with HyFOAM. Shown below are a series of overlaid results of the work performed by Hyslop [17] with those from HyFOAM. These values where taken from the lower boundary layer / wall directly post the end position of the injector. This is to replicate the readings of the pressure sensors used in the real world test cases. These sensors can be seen visualized in figure 3.1 and labeled as pressure transducers. The values are only being represented from from the lower wall pressure transducers as the cases that experienced real world testing only had those values shown.

78 mm Injector and 0 mm Post Injector Length

The first shown case is with the configuration of on no post injector length and with the long injector at 78 mm. The results of this particular case has the highest noise threshold but the values still fall inline with the real world experimental non-combustion data. To resolve the
noise in the data a longer simulation period would be desired. Though that being stated, the results still show good replication to that of the real world data and some of the trends seen in the CFD+ generated data.

Figure 3.2: Overlaid results from 78 mm Injector and 0 mm Post Injector Length configuration.

Figure 3.3: Mach, Temperature, and Velocity CFD results for 78 mm Injector and 0 mm Post Injector Length configuration.

**78 mm Injector and 50 mm Post Injector Length**

This test case, with the long injector and 50 mm post injector length, began the trend towards better replication of experimental data then the CFD+ results were capable of. As
can be seen starting around the 0.34 m mark of the engine length, the fit of the HyFOAM results to that of the non-combustion experimental data is near perfect. The CFD+ results do not replicate the pressure spike in that region at all.

Figure 3.4: Overlaid results from 78 mm Injector and 50 mm Post Injector Length configuration.

Figure 3.5: Mach, Temperature, and Velocity CFD results for 78 mm Injector and 50 mm Post Injector Length configuration.

48 mm Injector and 117 mm Post Injector Length

The final tested example is the short injector, at 48 mm, and the longest post injector length at 117 mm. As can be seen in figure 3.6, the HyFOAM results show nearly all flow trends
of that of the experimental non-combustion data. It was with this specific result that it was felt that proper validation of the HyFOAM software and workflow methodology was established. This then led into section 3.2 where a far more modern, more comparable to prior work accomplished with CReSS, and more overall complicated example of a scramjet engine which contained a combustor with a cavity based flame holder was found and tested.

Figure 3.6: Overlaid results from 48 mm Injector and 117 mm Post Injector Length configuration.

Figure 3.7: Mach, Temperature, and Velocity CFD results for 48 mm Injector and 117 mm Post Injector Length configuration.
3.2 Test Case 2

Developing the ability to appropriately model scramjet like simulations is paramount to accessing the validity of the work performed. Because of this, a real world example was chosen in the 2011 paper Experimental Characterization of Hydrocarbon-Fueled, Axisymmetric, Scramjet Combustor Flowpaths [14] as it contained a simple geometry which could be imported into CReSS and a Tare case which would allow for early CFD modeling without the need to model combustion. To accomplish this the decision was made to use the ‘Divergent’ [14] geometry, shown in figure 3.9 with inlet based area ratios shown in table 3.2, as it would allow for a smoother transition to CReSS while still retaining the singular cavity based combustor volume[10]. Meshing of this geometry is discussed in further detail in section 2.4 as it forced drastic workflow changes due to the included cavity.

<table>
<thead>
<tr>
<th>Table 3.2: Combustor area ratios for Step and Divergent scramjet geometries [14].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
</tr>
<tr>
<td>Divergent</td>
</tr>
</tbody>
</table>
3.2.1 Case Setup

From this test case the point of focus came upon the tare case as it allowed for rapid testing of both HyFOAM and Kestrel without the required learning curve and extensive simulation time of combustion modeling. In table 3.3, the bold and italicized values are the inlet values used in modeling the case. The inlet Mach is declared as 3.0 which results in the tested scramjet post inlet Mach being Mach 1.8. The walls of the scramjet where simulated as no slip and adiabatic. These selections for the wall where found to be adequate for modeling the tare case. The nozzle was set as an outflow in HyFOAM and in Kestrel it was declared as a sink which had an extrapolated back pressure. The declared boundary identifications for the tare case can be seen in figure 3.10 and differ from a combustion equivalent only in the lack of declared injectors.

Table 3.3: Combustor operating conditions for experimental runs [14].

<table>
<thead>
<tr>
<th>$q_0$ (psf)</th>
<th>M = 1.8 Facility Nozzle</th>
<th>M = 2.2 Facility Nozzle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>55 psia 1250 R</td>
<td>55 psia 1500 R</td>
</tr>
<tr>
<td>750</td>
<td>42 psia 1500 R</td>
<td>80 psia 1950 R</td>
</tr>
</tbody>
</table>
3.2. Test Case 2

3.2.2 CFD Solver Limitations

During the testing of this validation case with HyFOAM an issue began to arise in the form of points of extreme values forming in the isolators flow irregardless of mesh construction. These points, shown clearly as pressure in figure 3.11a, ranged across all simulated values and where random in nature. Figure 3.11b displays one of the tested meshes which showed these artifacts. These artifacts could range many of factors ten negative to positive depending on the values being viewed, namely temperature and velocity would display the largest deltas, and would ultimately spiral the simulations out of control. Refactoring the mesh did not solve the issue presented as it occurred irregardless of mesh aspect ratio, cell size, and cell type. Due to this, and the struggles with working with the undocumented combustion solver and lack of effective communication with HyFOAM’s developer, HyFOAM was declared
to not be fit for usage with this project. Kestrel became the sole point of focus for CFD simulations because of this. The artifacts do not occur in Kestrel simulations when using the same meshes.

Figure 3.11: Extreme pressure values shown to be forming within isolator section of scramjet without a. and with b. mesh shown.
3.2.3 Case Results

Shown in figure 3.12 with the results from the Kestrel based simulations in red; effective matching to the real world experimental test data, shown in blue, was achieved. The Kestrel’s results where scaled to match the test cases’ imperial units using the ratio shown in equation 3.1. One point of note is the ‘bump’ in the relatively smooth data points at around 10 x/D; this position coincides to combustor volumes cavity. A visual of the flow through the scramjet can be seen in figure A.3. In said figure, the Mach flow shows a higher velocity section towards the backsplash of the cavity volume in comparison to the rest of the cavity. Though not as visible in the pressure visualization of the same figure, there is a higher pressure value which is the root cause of the spike seen in figure 3.12.

\[
1 \frac{lbfs}{lbm} \approx 9.8 \frac{Ns}{kg}
\]  

(3.1)

![Figure 3.12: Overlaid values of mass flux scaled tare case results from the experimental data to that of the generated data from Kestrel. [14]](image)
Chapter 4

Testing Methodology

To test the concept of the morphing combustor to its maximum it was required that the conceptual sizing of the combustor cavity must be to simple controlling values and the external testing ranges be in depth but not overtly complex. To accomplish this, the cavity’s geometry, discussed in more detail later on in section 4.1, was reduced to a simplistic version, as shown in figure 4.1. This was to more easily express the motion a single, simple action by which could be replicated in the real world at a later date. Another required effect was the braking down of the required inputs of both CReSS and the CFD simulations at the beginning of the isolator stage of the scramjet engine. The focus was set to only Mach, Pressure, and Temperature at that stage of the engine. To further simplify this the pressure and temperature where determined by external altitude values, as derived in section 4.2, so the final point of focus could be just Altitude versus Mach. A fortunate benefit of this method of brake down of the inputs is that the testing methodology used in CReSS directly translates to that of the CFD solvers.

![Figure 4.1: Variables in Geometry Selection; L is Length of cavity upper, H is the Depth of the cavity referencing off of the throat to the cavity, and R is the Angle of the ramped rear wall.](image)

Figure 4.1: Variables in Geometry Selection; L is Length of cavity upper, H is the Depth of the cavity referencing off of the throat to the cavity, and R is the Angle of the ramped rear wall.
4.1 Engine Geometry

The variable geometry engine has three main adjustable attributes; namely the depth of the chamber, the angle of the chambers ramped rear wall, and the chambers upper length as seen in Fig: 4.2b. For this project only the length of the cavity was explored. For the depth of the cavity 0.021m was selected as it was found to have an average unstart rate of 32% which was 8% less than the average unstart rate of other simulated depths, ranging from 0.0029m to 0.021m with 5,700 CReSS based simulations. The rear radius was set to $159^\circ$ as this very close to the HiFiRE rear back splash angle. This was chosen to make the morphing combustor cavity volumes as reasonably close to the HiFiRE’s combustor cavity but retain the ability to have a single linear movement actuate the volume. The lengths were chosen, being 0.025m, 0.065m, and 0.11m, as shown in Fig: 4.3, as the selection of lengths representing an encompassing range around the HiFiRE geometry. Though certainly not a fine covering of all possible lengths, the selected lengths allow for rapid understanding of where the morphing engine should be positioned for highest performance. This concept would then be used to interpolate the engines combustor cavity to best match the speed and altitude of the engine in flight.

Figure 4.2: Simplified diagram of the combustor at different points of cavity volume with (a) being nearly fully expanded, (b) being at midpoint of travel, and (c) nearing the end of morphing range. [31]
Figure 4.3: Cross-sections of the HiFiRE geometry (a) and the different stages of the morphing combustor: 0.025m (b), 0.065m (c), and 0.11m (d). [31]

4.2 Required Inputs

The altitude and speed range tested were determined by using the constant freestream chart shown in Fig: 4.4b. The range being from 40 to 120 kft and from Mach 2.5 to 11.5. This range was selected as it was deemed to effectively test a variety of extreme values whose results can be seen in Fig: A.1. Further, it would cover the operational ranges of a ramjet and scramjet but go far enough that either/or would not function. Through just modulation of the combustor cavity near full coverage was desired.

\[
\frac{m_2}{m_0} = 1 - (2.5(1 - \mu_{KE}))^{\frac{1}{2}} \quad (4.1)
\]

\[
\frac{p_1}{p_0} = \frac{1}{\gamma + 1} (2\gamma m_0^2 - (\gamma - 1)) \quad (4.2)
\]

\[
\frac{p_2}{p_1} = \begin{cases} 
1 - 0.0776(m_0 - 1)^{1.35} & \text{if } m_0 < 5 \\
\frac{800}{m_0^2 + 935} & \text{if } m_0 \geq 5
\end{cases} \quad (4.3)
\]

\[
\frac{p_2}{p_0} = \frac{p_2}{p_1} \frac{p_1}{p_0} \quad (4.4)
\]
4.2. REQUIRED INPUTS

![Graph](image1)

(a)

![Graph](image2)

(b)

Figure 4.4: Correlation of input efficiency data [30] (a) and the standard day geometric altitude versus flight Mach number contours for constant freestream mass flow per unit area. A rough approximation of the range of Altitudes to Machs tested is marked by the red rectangle. [35] (b)

Similar to a CFD solver; for the CReSS solver to function, it must be given a geometry to test as well as the $m_2$ and $p_2$. This is due to CReSS only simulating from the exit of the inlet/ beginning of the isolator to the end of the nozzle. The derivation of $m_2$ is handled by Eqn: 4.1 derived from the University of Queensland’s paper on Scramjet inlets[30] with the $\mu_{KE}$ being set equal to 0.975 after referencing Fig: 4.4a. Doing so makes the formula equivalent to $\frac{m_2}{m_0} = \frac{1}{2}$. Unfortunately $p_2$’s derivation is not as straightforward beginning with Eqs: 4.5, 4.6, and 4.7. These formulae convert an inputted altitude in feet to pounds per square foot. This can then be converted to kPa using the ratio 1 kPa is equal to 20.89 psf. With the known $m_0$ the ratio of $\frac{p_2}{p_0}$ can be derived using Eqs: 4.2 [2], 4.3 [32], and 4.4. From all of this one can then plot out the value of $p_2$ due to the $m_0$ as seen in Fig: 4.5. These $m_2$ and $p_2$ values are then used with both the 1D friendly HiFiRE geometry and the generated morphing combustor positions.
Figure 4.5: P2 Pressure: Altitude Versus Inlet Mach

if Altitude < 36152 ft = \[
T = 59 - 0.00365A_{lt} \\
p_0 = 2116\left(\frac{T+459.7}{518.6}\right)^{5.256}
\]

(4.5)

if 82345 > Altitude ≥ 36152 ft = \[
T = -70 \\
p_0 = 473.1\epsilon^{1.73-0.000048A_{lt}}
\]

(4.6)

if Altitude ≥ 82345 ft = \[
T = -205.05 + 0.00164A_{lt} \\
p_0 = 51.97\left(\frac{T+459.7}{389.98}\right)^{-11.388}
\]

(4.7)

\[T_{mb} = \dot{m}_4u_4 - \dot{m}_2u_2 + (p_4A_4 - p_2A_2)\]  

(4.8)
Chapter 5

Results

By testing the different Mach speeds, different altitudes, and different combustor lengths, a developed concept of the performance possibilities apparent in the morphing combustor are levied. The CReSS based simulations are numbered in the thousands which allows for a medium resolution interpretation of the results. The CFD results total to 36 separate simulations as the computer hours to effectively model a combustion supporting hypersonic model are high, disregarding super computer based queue times. This flagrant time cost truly shows the great benefits a quasi-1d solver like CReSS provides. Though far less datum points then the CReSS based analysis, the 36 separate cases which are in itself only 9 separate points of comparison for Mach and Altitude when testing the three different combustor volumes and a HiFiRE geometry, the CFD based analysis should allow for an effective understanding of a CFD to CReSS relation in regards to expected results and possible unknown flow characteristics. Due to the individuality and resulting nature of the project, results are separated into respective groupings to better discuss individual findings from the different methods of simulation.

5.1 Quasi-1D

To compare to real world geometries, all of the tested morphing combustor geometries were compared to that of the 1D friendly HiFiRE geometry used in the development of CReSS[10].
Comparing the geometries tested reveals an interesting power curve (PC) like progression where the different geometries fall in and out of having the highest momentum imbalance. This imbalance can be seen as the performance of the engine as all momentum balance transitions are seen from stage 2 to stage 4 of the Scramjet engine; these stages being shown in Fig: 5.1. This was done as by not having the nozzle morph to match the modified combustor (i.e., only using stages 2-4 and not through stage 9) the stages beyond stage 4 provided inadequate results from the comparison of the engines. Performance figures from stage 2 to 9 showed that the HiFiRE engine, with matching nozzle, sporadically outperformed the generated morphing combustor positions whenever the inputted altitude and Mach made the nozzle incongruous. A morphing and matching nozzle was not within the scope of this project.

Comparing the Momentum Balance, found by using Eqn: 4.8, to the Altitude at a select Mach speed begins to show these PCs as shown in Fig: A.1. The PC of the various engine configurations has, on average, the test geometries out performing that of the HiFiRE. Importantly, one can see that the length of the test engines matter; not necessarily by large margins, but there are transitions where certain engines notably perform better. At Mach 7.5, one can see in figure 5.2c that at 60,000 ft the longest combustor outperform
the others yet just prior to that at 50,000 ft the longest combustor was well outperformed by the medium combustor. What this puts forth is the concept of tuning. Equivalent to that of tuning, any other combustion engine with the added ability to modify the physical parameters of the engine; not just that of fuel to air ratios. An aspect which can be tuned for is the temperatures seen within the combustor. This is where the HiFiRE geometry shows excellence with speeds past Mach 4.5 one can see in 5.3a that a clear and notable separation between the chamber performance of the morphing combustor engine and the HiFiRE’s. The HiFiRE’s engine capacity for heat control allows for its physical construction to not be under anywhere near the same level of duress seen by the generated morphing combustor. A complete collection of momentum balance versus altitude and temperature vs altitude charts.
can be seen in section A.1.

Figure 5.3: Selected Temperature Versus Altitude charts (lower is better) for Mach 4.5 (a) Mach 6 (b) Mach 7.5 (c) Mach 9 (d).

At Mach 7.5 in figure 5.2c the HiFiRE and the smallest morphing combustor position become negative at 90,000 feet. Then from Mach 8 to Mach 11.5 the momentum balance for any of the engine designs never goes positive again. This is interesting as the CReSS solver does not declare this situation as an unstart as the shock train length does not exceed the isolator length. This swing to negative momentum balance may be due to CReSS’s lack of flow recycling within the combustor; a point noted by the developer [10]. This can also be seen in the interesting development of the in chamber temperatures seen in figure 5.3d as at and above Mach 8 a temperature disparity of around 200 K can be seen between the HiFiRE geometry and the morphing combustor positions. A cause of this may be due to
the area of the stage 4 throat of the HiFiRE geometry being 243.84 mm$^2$ larger than that of the morphing combustor’s leading to less retention of in combustor cavity temperature. All of these points of interest are in a way revealed through the usage of the following CFD analysis as visual aspect of the results allow for in depth flow characterization.

Figure 5.4: Thrust Performance Maps: Stage 2 to Stage 4 without (a) and with HiFiRE (b), Stage 2 to Stage 9 without (c) and with HiFiRE (d). Decoder (e).
Figure 5.5: CFD results from combustion case simulation of the HiFiRE axis-symmetric alternative scramjet geometry showing a. Mach, b. Pressure, and c. Temperature.

5.2 CFD

Modeling of the morphing combustor in a CFD solver began with the usage of the HiFiRE axis-symmetric geometry with a simplified combustion model to replicate the ethylene mixture used in the real world testing and the CReSS simulations [18]. The chemistry model used is a 7 species, and 3 reaction model focused specifically on the replication of the flame speed, supplied to the project by Robert Decker. The flow results from the usage of this model can be seen in figure 5.5 which shows the Mach, Pressure, and Temperature of the model. It can be seen that the cavity interaction with this flow is minor due to simulation replicating an unstart event. The initial conditions for this flow replicate that of a freestream Mach of 7.8 with an altitude of around 19,000 ft. This case proved to be a great baseline
as it was designed to be configurable to all of the tested morphing combustor geometries while retaining important configuration requirements such as injector angles. The injectors interaction with the flow can be seen clearly in figure 5.5b just before the combustor cavity location.

Table 5.1: Momentum Balance (MB) and Temperature (T) values from CFD Results

<table>
<thead>
<tr>
<th>Mach</th>
<th>Altitudes [ft]</th>
<th>Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40,000</td>
<td>80,000</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>HiFire</td>
<td>Short</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>HiFire</td>
<td>Short</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>HiFire</td>
<td>Short</td>
</tr>
</tbody>
</table>

The nine separate positions by which are to be tested can be seen in table 5.1. As shown, the three altitudes tested are 40,000, 80,000, and 120,000 ft with the three external Machs being 3, 6, and 9. These are the external flow values where in simulation conditions must be converted to the stage two values using the formulations used by the CReSS simulations, seen in section 4.2. The choice of these initial conditions are twofold. First, the edge case tested ranges seek to justify the CReSS results for which either generated unstart scenarios or solver failure due to flow characteristics. The higher Mach range of tested values are used to help determine the flow conditions which CReSS experienced as negative momentum balance values but without a declared unstart event. These must be investigated as they determine the effectiveness of the results CReSS and what assumptions can be made off of the results garnered at these high altitudes and Mach speeds.
An example of the test cases to be run is shown in figure 5.6, where at an external Mach of 6 and altitude of 80,000 ft all of the morphing geometries and HiFiRE geometry were tested. For this scenario each cases was run where the activation energy requirements of the chemistry model were reduced so to force ignition in the scramjet models. After 2,000 iterations, these where to allow the flow to reach a steady state, the test cases then had the chemical model corrected to the appropriate values. This was then ran for another 2,000 iterations with the intention to allow for the models to settle once more. This was found to be not enough iterations to fully settle the models in either condition. This resulted in an unstart event. Further testing needs to allocate for longer simulation time in both setups to fully have the flow relax.

Figure 5.6: Mach values from CFD Results at external Mach of 6 and Altitude of 80,000 ft: HiFiRE (a), Small (b), Medium (c), and Long Morphing Combustor (d).
The mesh generation for the CFD simulations used by the model has a total cell count of around 17 million cells and contains a near wall initial offset of 50 micrometers. These values were chosen from Bornhoft et al. [5], where a very similar scramjet geometry was simulated with combustion modeling, as these values were found to be between the medium and fine mesh resolution parameters. Time constraints limited the ability to perform a full mesh resolution investigation but it was felt that the options chosen from the referenced work allowed for appropriate initial testing.
Chapter 6

Closing Remarks

6.1 Conclusions

The testing of various stages of a morphing combustor through the usage of the 1D Hypersonic solver CReSS has led to interesting revelations in the performance of engines with dedicated variable geometry combustors. Through the transition of the combustors length, the researchers were able to develop power curves where the point of high performance can be ascertained to a variety of atmospheric and operational situations. Furthermore, the comparisons to the proven HiFiRE geometry shows that the generated morphing combustor positions show promise and in many situations outperformed the HiFiRE engine. Poignantly, this opens the floodgates in regards to future work stemming from this project. Upping the resolution of the length values so as to get a finely focused view of how the combustor acts could lead to a well developed performance model that if combined with nozzle morphing or cavity throat optimization could prove highly effective. This is not to say the solution is one which is to supersede or replace that of morphing inlet or nozzle approach’s, but to be another approach by which an engine can be made more effective. This is just one simple mechanism that in combination with others can become a complicated and highly effective solution.
6.2 Future Work

While initial results show trends towards effective reactive scramjet engine combustor volumes there are a variety of both geometry decisions and other points of interest which would allow for the expansion of this project well into the future. Further the concepts expressed in regards to the morphing ability of the combustor cavity volume allots for the conceptional expansion to other portions of the scramjet engine namely the nozzle and inlet.

6.2.1 CReSS’ Continuous Improvements

CReSS, like many software, is constantly in a state of evolution where features and refinements are being developed to improve its results. In this project there where many cases where isolator conditions led to cases where in which CReSS completely failed to simulate and generated no data. This can be related to a multitude of different faculties of the code as this project acted like an extreme stress test as it pushed the boundaries of assumptions made in the formulas used in CReSS. Points of contention noted by its creator is the lack of combustor volume re-circulation and resulting forced assumptions of total enthalpy and total pressure [10].

Being that this projects main point of focus is the combustor volume these assumptions play into the results on display. For this, as CReSS is continuously developed, review in the form of rerunning a selection of cases tested in this project may be deemed necessary as new insights from the refinements can possible lead towards novel solutions that differ from the ones found here.
6.2.2 CReSS Modeling Expansion

A point of further modeling comparison so to refine result understanding and HiFiRE comparison effectiveness is in the test cases geometry of the morphing combustor. The morphing combustor cavity has the same diameter throat in the beginning of the cavity to that of the end of the cavity. This differs from the HiFiRE combustor cavity which experiences a large exit throat diameter to that of the entrance. This difference in the exit throat sizing is believed to in someways hamper the morphing combustors performance. To address this discrepancy, the generation script for the morphing combustor geometries can be modified such that the cavity exit throat diameter is set to match that of the HiFiRE geometry with a forced limit on the minimum height of the morphing combustor geometry being above the previously referenced value.

6.2.3 CFD Modeling Expansion

Due to the time cost and availability of capable systems a full replication of CReSS results with CFD modeling was not achieved. That is not to be said that the ground work had not been completed but that there is room to raise the resolution of CFD based results. The project is currently primed to have other researchers run and analyze the results for Kestrel based simulations of the variety of initial conditions tested with CReSS. This will take thousands of Machine hours to fully developed the model to appropriately create the performance mapping. From the data garnered mapping interpolations can determined so to create a truly capable solution which can be later expanded with added initial conditions. This process is near limitless in regards to depth of modeled inputs and reactionary measures. Points of focus can easily be exemplified by material temperature monitoring where the size of the cavity is varied to help manage the overall temperature.
6.2. Future Work

Another point where further data can be used to improve the resulting performance mapping is the inclusion of inlet modeling which is not currently possible within CReSS. Doing so would allow for the introduction of mainly turbulence interactions which the morphing combustor cavity should be found necessary to react to maintain combustion.

6.2.4 Moving Wall Modeling

The eventual point of focus for future work of this project is the modeling a moving wall in a live flow. This is a truly novel point of research with little to no comparable examples of internal scramjet movement. Major points of focus are thus: compression and expansion of the cavity flow reactions, possible forced unstart and/or restart events through compression based combustion reactions, wall movement speed interactions, wall movement force requirements, and temperature extremum location identification to name a few. All of these focal points are exciting as if any show effective flow interactions novel solutions can be derived.


[23] Tarun Mathur, Mark Gruber, Kevin Jackson, Jeff Donbar, Wayne Donaldson, Thomas


[31] Andrew L. Sorensen, Andrew B. DellaFera, Daniel Doyle, and Jonathan Black. Para-
BIBLIOGRAPHY


Appendices
Appendix A

Chart Repositories

A.1 CReSS Results Charts

A.2 CFD Flow Charts
A.2. CFD Flow Charts

Figure A.1: Momentum Balance Versus Altitude by Mach (a) Mach 2.5 (b) Mach 3 (c) Mach 3.5 (d) Mach 4 (e) Mach 4.5 (f) Mach 5 (g) Mach 5.5 (h) Mach 6 (i) Mach 6.5 (j) Mach 7 (k) Mach 7.5 (l) Mach 8 (m) Mach 8.5 (n) Mach 9 (o) Mach 9.5 (p) Mach 10 (q) Mach 10.5 (r) Mach 11 (s) Mach 11.5
Figure A.2: Combustor Chamber Temperature Versus Altitude by Mach (a) Mach 2.5 (b) Mach 3 (c) Mach 3.5 (d) Mach 4 (e) Mach 4.5 (f) Mach 5 (g) Mach 5.5 (h) Mach 6 (i) Mach 6.5 (j) Mach 7 (k) Mach 7.5 (l) Mach 8 (m) Mach 8.5 (n) Mach 9 (o) Mach 9.5 (p) Mach 10 (q) Mach 10.5 (r) Mach 11 (s) Mach 11.5
Figure A.3: CFD results from tare case simulation of tested scramjet geometry showing a. Mach, b. Pressure, and c. Temperature.
Appendix B

Code Repositories

B.1 CFD Code Repository

This is a collection of scripts and pieces of code used in the analysis of the data presented and the workflow performed.

B.1.1 Wall-Reader.py

```python
# Loads tap csv and pulls the wall values
# Andrew Sorensen - 08/09/2021

# Imports
import Tap_Reader as tp
import pandas as pd
import matplotlib.pyplot as plt
import grid_maker as gm

# Functions
# Pulls in the tap data,
def get_Data(name):
    data = pd.read_csv('taps_output/' + name + '.csv')
    return data
```
# Plots the cleaned data versus the paper

def plots(d1, d2):
    plt.figure()
    # Color the levels
    c1e = d1[(d1['Y'] >= 0.037) & (d1['Y'] <= 0.039)]
    c1 = d1[(d1['Y'] >= 0.037) & (d1['Y'] <= 0.039) & (d1['X'] <= 0.825)]
    c2e = d1[(d1['Y'] >= 0.050) & (d1['Y'] <= 0.051)]
    c2 = d1[(d1['Y'] >= 0.050) & (d1['Y'] <= 0.051) & (d1['X'] <= 1.320)]
    c3 = d1[(d1['Y'] >= 0.063) & (d1['Y'] <= 0.064) & (d1['X'] <= 1.625)]
    c4 = d1[(d1['Y'] >= 0.076) & (d1['Y'] <= 0.077) & (d1['X'] <= 1.912)]
    c5 = d1[(d1['Y'] >= 0.088) & (d1['Y'] <= 0.089) & (d1['X'] <= 2.080)]
    c6 = d1[(d1['Y'] >= 0.100) & (d1['Y'] <= 0.110) & (d1['X'] <= 2.130)]
    c7 = d1[(d1['Y'] >= 0.114) & (d1['Y'] <= 0.115) & (d1['X'] <= 2.200)]
    c8 = d1[(d1['Y'] >= 0.127) & (d1['Y'] <= 0.128) & (d1['X'] <= 2.247)]
    c9 = d1[(d1['Y'] >= 0.140) & (d1['Y'] <= 0.141) & (d1['X'] <= 2.310)]
    c10 = d1[(d1['Y'] >= 0.152)]

    #ax = d1.plot(x='x/D', y='P/RhoU', kind='scatter', c='red', label='Generated')
    ax = c1e.plot(x='x/D', y='P/RhoU', kind='scatter', c='red', label='Kestrel: Non-Combustion')
    #c2e.plot(x='x/D', y='P/RhoU', ax = ax, kind='scatter', c='orange', label = 'G - R2')
    # c3.plot(x='x/D', y='P/RhoU', ax = ax, kind='scatter', c='green', label='G - R3')
    # c4.plot(x='x/D', y='P/RhoU', ax = ax, kind='scatter', c='purple', label = 'G - R4')
    # c5.plot(x='x/D', y='P/RhoU', ax = ax, kind='scatter', c='aquamarine', label='G - R5')
    # c6.plot(x='x/D', y='P/RhoU', ax = ax, kind='scatter', c='0.8', label='G - R6')
APPENDIX B. CODE REPOSITORIES

```python
# c7.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', c='y', label='G − R7')
# c8.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', c='k', label='G − R8')
# c9.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', c='0.2', label='G − R9')
# c10.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', c='m', label='G − R10')
d2.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', label='Experiment: Non−Combustion')
# plt.title('Paper Comparison: P/RhoU')
plt.ylabel('P/RhoU [(lbf−s)/lbm]')
plt.legend()
plt.savefig('compCleaned.png')

diam = pd.read_csv('geoms/Gruber.csv')
diam = pd.concat([diam, pd.DataFrame(data={'Y2': 0.55 * (diam['Y']**2 + diam['Y'])*0.5})], axis=1)
print(diam)
ax = c1.plot(x='X', y='Y', kind='scatter', c='red', label='G − R1', xlim=[0, 2.4], ylim=[0, 0.24])
c2.plot(x='X', y='Y', ax=ax, kind='scatter', c='orange', label='G − R2',
xlim=[0, 2.4], ylim=[0, 0.24])
c3.plot(x='X', y='Y', ax=ax, kind='scatter', c='green', label='G − R3',
xlim=[0, 2.4], ylim=[0, 0.24])
c4.plot(x='X', y='Y', ax=ax, kind='scatter', c='purple', label='G − R4',
xlim=[0, 2.4], ylim=[0, 0.24])
c5.plot(x='X', y='Y', ax=ax, kind='scatter', c='aquamarine', label='G − R5',
xlim=[0, 2.4], ylim=[0, 0.24])
c6.plot(x='X', y='Y', ax=ax, kind='scatter', c='0.8', label='G − R6',
xlim=[0, 2.4], ylim=[0, 0.24])
```

B.1. CFD Code Repository

```python
69 c7.plot(x='X', y='Y', ax=ax, kind='scatter', c='y', label='G - R7', xlim=[0, 2.4], ylim=[0, 0.24])
70 c8.plot(x='X', y='Y', ax=ax, kind='scatter', c='k', label='G - R8', xlim=[0, 2.4], ylim=[0, 0.24])
71 c9.plot(x='X', y='Y', ax=ax, kind='scatter', c='0.2', label='G - R9', xlim=[0, 2.4], ylim=[0, 0.24])
72 c10.plot(x='X', y='Y', ax=ax, kind='scatter', c='m', label='G - R10', xlim=[0, 2.4], ylim=[0, 0.24])
73 diam.plot(x='X', y='Y2', ax=ax, kind='line', label='Geometry', xlim=[0, 2.4], ylim=[0, 0.24])
74 plt.title('Tap Points')
75 plt.legend()
76 plt.savefig('sideShot.png')
77
78 # Main
79 def main(name):
80     dtc = get_Data(name)
81     #dead = dtc[dtc['Found'].isin([0])]
82     live = dtc[dtc['Found'].isin([1])]
83     live = pd.concat([live, tp.get_XD_V2(live)], axis=1)
84     live = pd.concat([live, tp.get_PRhoU(live)/9.8], axis=1)
85     live = live[(live['Y'] >= 0.028) & (live['P/RhoU'] < 100)]
86     print(live.head())
87     plots(live, tp.get_Gruber())
88
89 if __name__ == '__main__':
90     plt.close('all')
91     file = 'taps00015000'
92     main(file)
```
B.1.2 Tap-Reader.py

```python
# Imports csv files created by kestrel for tap locations and returns cleaned arrays
# Andrew Sorensen − 07/12/2021

# Imports
import pandas as pd
import matplotlib.pyplot as plt
import grid_maker as gm

# Functions

# Get the diameter for x/D
def get_diam(xPnt, dataSet):
    return (2*gm.interectDetect(dataSet,xPnt))

# Returns x/D
def get_XD(data,name):
    xD = []
    diam = []
    diam_Data = gm.injest(name)
    for d in data:
        gen = get_diam(d,diam_Data)
        diam.append(gen)
        xD.append(d/gen)
    return (pd.DataFrame(data={'Diameter':diam,'x/D':xD}))

# Returns x/D at the read point, not the outside... like an idiot
def get_XD_V2(dataSet):
```

```python
diam = pd.DataFrame(data={'Diameter': 2 * (dataSet['Y']**2 + dataSet['Z']**2) **0.5})
xD = pd.DataFrame(data={'x/D': dataSet['X'] / diam['Diameter']})
return xD

# P / (rho U) Ultimately its velocity but done in an odd manner
# This may need to have unit conversions applied
# 1 lbf - s / lbm = 9.806609844 pa / (kg/(m^2 s)) = 9.806609844 m/s ... Confusing

def get_PRhoU(pDV):
    return (pd.DataFrame(data={'P/RhoU': pDV['Pressure'] / (pDV['Density'] * pDV['VelocityMag'])}))

# Get the original Gruber paper results

def get_Gruber():
    name = 'Gruber_Paper'
    data = pd.read_csv(name+'.csv', names=['x/D', 'P/RhoU'], header=0)
    return(data)

# Plot the injected data versus papers data xlim=[-4,24], ylim=[0,120],
def plot_Comp(dat_1, dat_2, name):
    plt.figure()
    ax = dat_1.plot(x='x/D', y='P/RhoU', kind='scatter', c='red', label='Generated', xlim=[-4, 24], ylim=[0, 120],)
    dat_2.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', label='Paper',
               xlim=[-4, 24], ylim=[0, 120],)
    plt.title(name + ' Paper Comparison: P/RhoU')
    plt.legend()
    plt.savefig('compPRU.png')
    plt.figure()
```

74

```python
ax = dat_1.plot(x='x/D', y='P/RhoU', kind='scatter', c='red', label='
Generated')
dat_2.plot(x='x/D', y='P/RhoU', ax=ax, kind='scatter', label='Paper')
plt.title(name + ' Paper Comparison: P/RhoU Unlimited')
plt.legend()
plt.savefig('comp_NoLim.png')

def main(file, name):
data = pd.read_csv('taps_output/' + file + '.csv')
data = pd.concat([data, get_XD_V2(data)], axis=1)
data = pd.concat([data, get_PRhoU(data) / 9.8], axis=1)
data.to_csv('output.csv')
plot_Comp(data, get_Gruber(), name)

if __name__ == '__main__':
    plt.close('all')
    file = 'taps00015000'
    name = 'Gruber'
    main(file, name)
```

### B.1.3 Grid-Maker.py

```python
# Andrew Sorensen 7/9/2021
# Creates a tap file for the models. Very rudimentary and more less edited per
case.
# Works for what I am doing.

# Imports
import numpy as np
```
# Constants — Change as necessary

# 'Gruber' 'HiFiRE'

gename = 'Gruber' # Change this to match whatever engine you want taps for

xspc = 0.0127 # 0.5 in = 0.0127 m

yszpc = 0.0127

direc = -1 # 1 Means its a wedge in the pos x,y,z direc, -1 means x,y,-z

# Open a file and pulls the 2d approximation which then will be used to fill
# the tap points

def injest(filename):
    return np.loadtxt('geoms/' + filename + '.txt')

# mx+b line between the points.... Not perfect but it does work as an
# intersection detection

def lineBt(xy1, xy2, pt):
    m = (xy2[1] - xy1[1]) / (xy2[0] - xy1[0])
    b = xy2[1] - (m * xy2[0])
    return (m * pt + b)

# Returns the maximum Y value for that x point

def intersectDetect(data, xPnt):
    xy1 = 0
    xy2 = 0
    ind = 0
    for i in data:
        if xPnt >= i[0]:
            xy1 = i
            xy2 = data[ind+1]
            #sys.exit()
            exit
            ind = ind+1
APPENDIX B. CODE REPOSITORIES

```python
    return(lineBt(xy1,xy2,xPnt))

# Creates a list of points for a series of x values.
# Does it on a 45 degree angle for now, can be changed in the future
def dataGen(data):
    maxX = data[-1][0]
x = 0
taps = []
while x<maxX:
    y = 0
    maxY = interactDetect(data,x)
    while y<=maxY:
        taps.append([round(x,6),round(y,6),round(y*direc,6)])
y+=ySpc
x+=xSpc
return taps

# Writes to the specified file
def writeMain(name,data):
    f = open('taps/' + name + '_base.tap', 'w')
f.write('Taps\nTolerance\n'+str(len(data))+\n\t1E-5\n<Tap Locations\nfor i in data:
    f.write(\t'.join(map(str,i)))
    f.write(\n')
f.close()

# Main functions which combines all of the above functions
def main(name):
    data = injest(name)
taps = dataGen(data)
writeMain(name,taps)
```
### B.1.4 Hyslop-Comparison.py

```python
#!/usr/lib/python3

# Andrew Sorensen 100520
# Graphs the HyFOAM outputs to compare to Hyslop paper

# Imports
import matplotlib.pyplot as plt
import csv

# Paper Stuff Digitized
pcc = open('48_117_CFD_C.txt', 'r')
pccF = pcc.read().split('
')
pccX = []
pccY = []
for p in pccF[:-2]:
    tem = p.split(',', ' ')
    pccX.append(float(tem[0]))
    pccY.append(float(tem[1]))

pcn = open('48_117_CFD_NonC.txt', 'r')
pcnF = pcn.read().split('
')
pcnX = []
pcnY = []
for p in pcnF[:-2]:
```

```
tem = p.split(', ')  
pcnX.append(float(tem[0]))  
pcnY.append(float(tem[1]))  

pec = open('48_117_E_C.txt', 'r')  
pecF = pec.read().split('
')  
pecX = []  
pecY = []  
for p in pecF[:-2]:  	em = p.split(', ')  
pecX.append(float(tem[0]))  
pecY.append(float(tem[1]))  

pen = open('48_117_E_NonC.txt', 'r')  
penF = pen.read().split('
')  
penX = []  
peny = []  
for p in penF[:-2]:  	em = p.split(', ')  
penX.append(float(tem[0]))  
peny.append(float(tem[1]))  

# HyFoam gen stuff  
xu = []  
x = []  
t = []  
p = []  
rho = []  
u = []  

u1File = 'line_U1.csv'
t1prhoFile = 'line_Tt_p_rhof1.csv'
u2File = 'line_U2.csv'
t2prhoFile = 'line_Tt_p_rhof2.csv'

with open(u1File, newline='
') as fi:
    sr = csv.reader(fi, delimiter=" ", quotechar='|')
    for row in sr:
        xu.append(float(row[0]))
        u.append(float(row[1][1 :]))

with open(u2File, newline='
') as fi:
    sr = csv.reader(fi, delimiter=" ", quotechar='|')
    for row in sr:
        xu.append(float(row[0]))
        u.append(float(row[1][1 :]))

with open(t1prhoFile, newline='
') as fi:
    sr = csv.reader(fi, delimiter=" ", quotechar='|')
    for row in sr:
        xt.append(float(row[0]))
        t.append(float(row[1][1 :]))
        p.append(float(row[2][1 :]))
        rho.append(float(row[3][1 :]))

with open(t2prhoFile, newline='
') as fi:
    sr = csv.reader(fi, delimiter=" ", quotechar='|')
    for row in sr:
        xt.append(float(row[0]))
        t.append(float(row[1][1 :]))
        p.append(float(row[2][1 :]))
        rho.append(float(row[3][1 :]))

# Plots
fig, ax = plt.subplots()
fig.set_size_inches((12, 5))
#ax.plot(xt, t, label='T(t)')
ax.plot(xt, p, label='HyFOAM: Non–Combustion')
ax.plot(pccX, pccY, label='CFD+: Combustion')
ax.plot(pcnX, pcnY, label='CFD+: Non–Combustion')
ax.plot(pecX, pecY, label='Experiment: Combustion', linestyle='--', marker='o')
ax.plot(penX, penY, label='Experiment: Non–Combustion', linestyle='--', marker='D')
#ax.plot(xt, rho, label='rho')
#ax.plot(xu, u, label='U')
#ax.set_title('Pressure along Engine: 48mm Injector and 117mm Post–Injector Length')
ax.set_ylabel('Pressure [Pa]')
ax.set_xlabel('Distance along Engine [m]')
ax.set_xlim((0, .6))
ax.set_ylim((0, 200000))
plt.legend()
plt.show()
\textbf{B.2.1 Morphing-Combustor.py}

\begin{verbatim}
# Main program file which generates, runs, and sorts test cases
# Andrew Sorensen

import os
import shutil as sh
import fileGen as fG
from subprocess import call
import MandP_MM as mp2

def preSetup():
    print('Setting up file structure')
    try: os.mkdir(os.getcwd() + '/MM_Tests_50kPaP2')
    except: print('Main folder already existing."
    try: os.mkdir(os.getcwd() + '/Phase2_MM_50kPaP2')
    except: print('Graphical Directory already existing.')
    print('File structure generated.')

def testRun50(length, height, rad, Mach, alt):
    # File names
    name = 'MM_Geo\n'    direc = os.getcwd()
    storagefolder = direc + '/MM_Tests_50kPaP2'
    # measurements in meters
    # length = 0.043
    # height = 0.0171
    # rad = 21*(3.14/180)
    fG.main(name, length, height, rad)
    mp2.refFG(Mach, alt, 'FlightParam')
\end{verbatim}
```python
test1 = [[0,1],[22,2]]
test1.append([2,3])
print(fG.main(name, length, height, rad))
call(['python3', 'SSP2MM.py'])

#Gen files
rr = 'l' + str(length)[0:6] + ':h' + str(height)[0:6] + ':r' + str(rad)[0:6] + ':m' + str(Mach) + ':alt' + str(alt) #returned results
try: os.mkdir(storagefolder + '/' + rr)
except: print('Unable to create working directory')

#Move csv file
ocsv = direc + '/Phase2DataMM.csv'
ncsv = direc + '/' + rr + '.csv'
try: os.rename(ocsv, ncsv)
except: print('No generated .csv file found')
try: sh.move(ncsv, storagefolder + '/' + rr + '/' + rr + '.csv')
except: print('Unable to move csv file.')

#move geom file
geom = direc + '/' + name + '.xlsx'
try: sh.move(geom, storagefolder + '/' + rr + '/' + rr + '.xlsx')
except: print('Unable to move Geom file.')

#move graphs
gphs = direc + '/Phase2_MM_50kPaP2'
try:
    sh.move(gphs, storagefolder + '/' + rr + '/' + rr + 'Graphs')
os.mkdir(direc + '/Phase2_MM_50kPaP2')
except: print('Unable to move Graphs.')

#Move Flight Params
fp = direc + '/FlightParam.xlsx'
try:
    sh.move(fp, storagefolder + '/' + rr + '/' + rr + 'FP.xlsx')
```

except: print('Unable to move Flight Params.')

# Move Unsart or start decider
sus = direc + '/start-unstart.txt'

try:
    sh.move(sus, storagefolder + '/'+rr+'/'+rr+'SUS.txt')
except: print('Unable to move Start Unstart Decision.')

print('All possible files moved to folder: ' + rr + ' in /MM_Tests_50kPaP2 .')

#This one works??? Somehow . . . .
#length = 0.043
#height = 0.0171
#rad = 21*(3.14/180) = 0.3663
def main():
    preSetup()
    Mach = 2.5 #Max to 12
    alt = 30000 #Max to 140000 ft
    length = 0.105 #0.100
    height = 0.021
    rad = 21*(3.14/180) # DO NOT CHANGE .... FOR NOW
    for l in range(1,10): # (1,20)
        for h in range(1,10):#15
            for m in range(1,20):#35
                for a in range(1,11):
                    testRun50(length, height, rad, Mach, alt)
                    alt += 10000
                    Mach += 0.5
                    alt = 30000
                    height -= 0.002
                    Mach = 2.5
B.2.2 fileGen.py

```python
#!/usr/bin/env python2
#
# -*- coding: utf-8 -*-
""
Created on Mon Oct 28 12:10:28 2019
Generates input files for morphing combustor testing.
@author: andrew sorensen
""

import xlsxwriter as xw
import math

#Input name as string, length as float, height as float, rad as float radians
def main(name, length, height, rad):
    row = 0
    col = 0
    xang = height/math.tan(rad)
    #yang = math.sin(rad)
    statInd = (['Station','index'], ['B1',0], ['B2',1], ['I1',2],
```

length = 0.01
height = 0.021
print('Done')
main()
xLoc = ( ['Xlocation'], [0], [0.203], [0.244], [0.295], [0.296], [0.295 + length],
        [0.295 + length + xang], [0.295 + length + xang + 0.31],
        [0.295 + length + xang + 0.899])

yLoc = ( ['Height'], [0.0127], [0.0127], [0.0136], [0.0148], [0.0148 + height],
        [0.0148 + height], [0.0148], [0.0218], [0.1518])

area = ([['Areas']])
thick = ([['thickness']])
dept = ([['Depth']])
for x in range(1,10):
    area.append([[yLoc[x][0]*0.1016]])
    thick.append([[0.02]])
    depth.append([[0.1016]])
#time to make the generated geometry
wb = xw.Workbook(name + '.xlsx')
ws = wb.add_worksheet('Geom')
for a, b in statInd:
    ws.write(row, col, a)
    ws.write(row, col + 1, b)
    row+=1
row = 0
col = 2
for x in range(0,10):
    ws.write(row, col, xLoc[x][0])
    ws.write(row, col + 1, depth[x][0])
B.2.3 MandP-MM.py

```python
# coding: utf-8

""
Created on Thu Dec 5 11:59:22 2019

@author: andrew sorensen

import math
import xlsxwriter as xw

# This is going to convert inlet Mach to M2 Mach, Same goes for pressure and
```
#other values...

gamma = 1.36  #perfect gas 1.4 — Andy Recommends 1.36

#Input: Flight Object Mach
#Output: Combustor entrance Mach

def m0m2(m0):
    nuKE = 0.975  #m2/m0 = 0.5
    m2 = m0*(1-(2.5*(1-nuKE))**0.25)#Post inlet Mach - UQueensland Inlets
    return(m2)

#Input: Flight Object Mach
#Output: Combustor/Isolator entrance Pressure Ratio

def m0p2(m0):
    if m0 < 5:
        p2p1 = 1- 0.0776*((m0 - 1)**1.35)
    else:
        p2p1 = 800 / (m0**4 + 935)
    return(p2p1)

def p1p0(m0):
    return((1/(gamma + 1)) * (2 * gamma * (m0**2) - (gamma - 1)))

def p2p0(m0):
    return(p1p0(m0) * m0p2(m0))

#Input: Flight Object Mach, Alititude of Object (ft)
#Ouptut: Pressure (kPa) and Temperature (K) at Combustor Entrance

def altTP(m0, alt):
    T = 0  #Temp of inlet air
    P = 0  #p2
if (alt < 36152):
    T = 59 - 0.00365*alt
    P = 2116*((T + 459.7)/518.6)**5.256
if (alt >= 36152 and alt < 82345):
    T = -70
    P = 473.1*math.exp(1.73 - 0.000048*alt)
if (alt >= 82345):
    T = -205.05 + 0.00164*alt
    P = 51.97*((T + 459.7)/389.98)**-11.388
    #P = ((((101325*((1 -(( 2.25577*(10**(-5))) * (alt / 3.281))))**5.25588)))
    #/0.04788)*(1/1000)
    return([(P*0.04788)*p2p0(m0), (T-32)*(5/9)+273.15])

#Input: Flight Object Mach, Altitude of Object (ft)
#Output: Mach and Pressure (kPa) at Combustor Entrance
def m2p2(m0, alt):
    m2 = m0*m2(m0)
    p2 = altTP(m0, alt)[0]
    return(m2,p2)

#generate a reference file to which can be referenced by CRESS for Flight Param
#Use name 'FlightParam' or you will need to mod SSP2MM.py line 34
def refFG(m0, alt, name):
    m2, p2 = m2p2(m0, alt)
    struct = (["M2 Mach","P2 Pressure","Altitude"],[m2, p2, alt])
    wb = xw.Workbook(name + ".xlsx")
    ws = wb.add_worksheet(name)
    row = 0
    col = 0
    for a, b, c in struct:
ws.write(row, col, a)
ws.write(row, col + 1, b)
ws.write(row, col + 2, c)
row += 1
wb.close()
print('Flight Parameter File Generated. ')

def PMChart(Machs, alts, name):
    wb = xw.Workbook(name + '.xlsx')
    wp = wb.add_worksheet('Pressure')
    wp.write(0, 0, 'P2 Pressure')
    wm = wb.add_worksheet('Mach')
    wm.write(0, 0, 'M2 Mach')
    i = 1
    for m in Machs:
        wp.write(0, i, m)
        wm.write(0, i, m)
        i += 1
    i = 1
    for a in alts:
        wp.write(i, 0, a)
        wm.write(i, 0, a)
        i += 1
    row = 1
    col = 1
    for m in Machs:
        for a in alts:
            m2, p2 = m2p2(m, a)
            wp.write(row, col, p2)
            wm.write(row, col, m2)
            row += 1
row = 1
col +=1
wb.close()
print('Done')

mac = [2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5]
alts = [40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000]
name = 'PM'
PMChart(mac, alts, name)

B.2.4 MM-Aggregate.py

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
""
Created on Wed Jan  8 15:36:28 2020

@author: andrew sorensen
""

#Searching for failure
import os
import xlsxwriter as xw
import MM_csvread as cr
import math

""
Returns free stream pressure from altitude
""
def alt2P(alt):
    P = 0
    if (alt < 36152):
        T = 59 - 0.00365*alt
        P = 2116*((T + 459.7)/518.6)**5.256
    if (alt >= 36152 and alt < 82345):
        P = 473.1*math.exp(1.73 - 0.000048*alt)
    if (alt >= 82345):
        T = -205.05 + 0.00164*alt
        P = 51.97*((T + 459.7)/389.98)**-11.388
    return(P*0.04788)

def name2P(name):
    liz = name.split(':')
    alt = int(liz[4][3:])
    return(alt2P(alt))

def wsDataIn(worksheet, data, names):
    col = 0
    worksheet.write(0, col, names[0])
    worksheet.write(0, col + 1, names[1])
    worksheet.write(0, col + 2, names[3])
    worksheet.write(0, col + 3, names[4])
    worksheet.write(0, col + 4, names[5])
    row = 1
    for l, h, r, m, a, y in (data):
        worksheet.write(row, col, float(l))
        worksheet.write(row, col + 1, float(h))
        worksheet.write(row, col + 2, float(m))
        worksheet.write(row, col + 3, int(a))
        worksheet.write(row, col + 4, y)
APPENDIX B. CODE REPOSITORIES

row += 1

' ' ' ' ' ' ' ' ' ' ' ' ' ' 'Must run after wsdataatin

def wsDataAdd(worksheet, selection, directory, filenames):
    #filenames = os.listdir(directory)
    tfile = directory + '/' + filenames[0] + '/' + filenames[0] + '.csv'
    tplist = cr.csvseldata(tfile, selection)[0]
    i = 5
    for p in tplist:
        worksheet.write(0, i, p + 'c')
        worksheet.write(0, i+1, p + 'n')
        i += 2
    worksheet.write(0, i, 'Thrust 2->4')
    worksheet.write(0, i+1, 'Thrust 2->9')
    row = 1
    for x in filenames:
        file = directory + '/' + x + '/' + x + '.csv'
        try:
            data = cr.csvseldata(file, selection)
            #Data: 0 Name: 1 Chm: 2 Ext: 3 Init: 4 P4
            col = 5
            for y in range(0, len(selection)):
                worksheet.write(row, col, float(data[1][y]))
                worksheet.write(row, col+1, float(data[2][y]))
                col += 2
            t1 = float(data[4][5]) * float(data[4][6]) #5: mdot, 6: Velocity
            t2 = float(data[3][5]) * float(data[3][6]) #data 3 init
            t3 = ((float(data[4][0]) * 0.00150368) - (float(data[3][0]) * 0.00129032))#name2P(x)
worksheets.write(row, col, t1 - t2 + t3)

t1l = float(data[2][5]) * float(data[2][6]) #5: mdot, 6: Velocity

t2l = float(data[3][5]) * float(data[3][6]) #data 3 init

t3l = ((float(data[2][0]) * 0.00150368) - (float(data[3][0]) * 0.00129032)) #name2P(x)

worksheet.write(row, col + 1, t1l - t2l + t3l)

row += 1

except:

col = 5

for y in range(0, len(selection)):
    worksheet.write(row, col, 'Missing CSV')
    worksheet.write(row, col + 1, 'Missing CSV')

    col += 2

    worksheet.write(row, col, 'Missing CSV')

row += 1

return 0

def lNames(names, run):
    liz = []

    for x in names:
        vals = x.split(': ')
        liz.append([vals[0][1:], vals[1][1:], vals[2][1:], vals[3][1:], vals[4][3:], run])

    return liz

#Dataset selection

direc = os.getcwd() + '/U_S'
tpLn = (["Length", "Height", "Angle", "Mach", "Altitude", "Runs?"])

St = direc + '/Starts'
namesSt = os.listdir(St)

starts = lNames(namesSt, 'Y')
Ut = direc + '/UnStarts'
namesUt = os.listdir(Ut)
unstarts = lNames(namesUt, 'N')
Nf = direc + '/NoFile'
namesNf = os.listdir(Nf)
nofile = lNames(namesNf, 'N/A')

All = []
All.extend(starts)
All.extend(unstarts)
All.extend(nofile)

dataset = [11,7,8,3,5,4,6] #MM_cvsread for translation

#Making the Book
book = xw.Workbook('DataAnalysis.xlsx')
wsA = book.add_worksheet('All')
wsDataIn(wsA, All, tpLn)
wsT = book.add_worksheet('Starts')
wsDataIn(wsT, starts, tpLn)
wsDataAdd(wsT, dataset, St, namesSt)
wuT = book.add_worksheet('UnStarts')
wDataIn(wuT, unstarts, tpLn)
wDataAdd(wuT, dataset, Ut, namesUt)
wfF = book.add_worksheet('No Files')
wsDataIn(wfF, nofile, tpLn)

book.close()
print('Done')
B.2.5 MM-csvread.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

Created on Fri Jan 10 15:17:33 2020

@author: andrew sorenson

import csv
# import os

""
Returns the numbers of rows of the CSV file
""

def csvsum(filename):
    with open(filename) as csvfile:
        readCSV = csv.reader(csvfile, delimiter=',',)
        row_count = sum(1 for row in readCSV)
    return(row_count)

""
Pulls the names and from lines 28 and the last line and returns unfocused data

""

def csvdtrt(file):
    end = csvsum(file) - 1
    name = []
    init = []
    chm = []
   cmc = []
    ext = []
```

with open(file) as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',
) i = 0
    for row in readCSV:
        if i == 0:
            name.append(row)
        if i == 1:
            init.append(row)
        if i == 41:
            chm.append(row)
        if i == 50:
           cmc.append(row)
        if i == end:
            ext.append(row)
        i+=1
    return (name, chm, ext, init,cmc)

def csvsel(dat(filename, selection):
    data = csvdt(filename)
    ret = []
    i = 0
    for n in range(0,5):
        i = 0
        tmp = []
        for x in data[n][0]:
            tmp.append(x)
if i in selection:
    tmp.append(x)
    i += 1
    ret.append(tmp)
return(ret)

#direc = os.getcwd() + '/U_S/Starts/10.06:h0.02:r0.3663:m3.0:alt90000'
#filename = '10.06:h0.02:r0.3663:m3.0:alt90000.csv'
#ff = direc + '/' + filename
#print(csvseldat(ff,[11,7,8,3,5,4,6]))

B.2.6 MM-Data-Analysis.py

#!/usr/bin/env python3
#
# coding: utf-8
#
""
Created on Tue Nov 12 15:23:45 2019

@""

import os
import shutil

testdir = os.getcwd() + '/MM_Tests_50kPaP2'

This determines wether there was a start or unstart detected.
Exceptions made for files which did not generate a file. ie NoFile
def unStartQ(name):
    try:
        file = open(testdir + '/' + name + '/' + name + 'SUS.txt', 'r')
        us = int(file.read())
        file.close()
    except:
        us = 2
        print('No Start-Unstart file with test: ' + name)
        #print(us)
    return us

'Makes storage directories.'

'def sepfl():#setup files
    try:os.mkdir(os.getcwd() + '/U_S/Starts')
    except:print('Start Directory Already Existing.')
    try:os.mkdir(os.getcwd() + '/U_S/Unstarts')
    except:print('UnStart Directory Already Existing.')
    try:os.mkdir(os.getcwd() + '/U_S/NoFile')
    except:print('NoFile Directory Already Existing.')

'Copies files to correct folders.'

def main():
    names = os.listdir(testdir)
    sepfl()
    start = []
    unstart = []
    for x in names:
49  src = testdir + '/ ' + x
50  dest = os.getcwd() + '/U_S'
51  ww = unStartQ(x)
52  if (ww == 0):
53      destS = dest + '/Starts/' + x
54      shutil.copytree(src, destS)
55      start.append(x)
56  if (ww == 1):
57      destU = dest + '/UnStarts/' + x
58      shutil.copytree(src, destU)
59      unstart.append(x)
60  if (ww == 2):
61      destN = dest + '/NoFile/' + x
62      shutil.copytree(src, destN)
63      unstart.append(x)
64  print('Done and files copied.')
65
66 main()

B.2.7 Morph-Contour.py

1  #!/usr/lib/python3
2
3  # Make some cool charts my dude!
4  # Andrew Sorensen : 24−08−20
5
6  # imports
7  import os
8  import pandas as pd
9  import matplotlib.cm as cm
import matplotlib.pyplot as plt
import numpy as np

# Read the csv files
def fileOpen(name):
    file = pd.read_csv(name + '.csv')
    wanted = ['A', 'M', 'mdot', 'u', 'Ps', 'Pt', 'Ts', 'Tt']
    return file[wanted]

def search(data, x, y, itr):
    # This is inefficient
    for d in data:
        if ((x in d) and (y in d)):
            if d[itr] >= 0: return (d[itr])
            else: return (0)
    else: continue
    return (0)

def getZ(Mach, alt, data, itr):
    ret = []
    for a in alt:
        tmp = []
        for m in Mach:
            tmp.append(search(data, m, a, itr))
        ret.append(tmp)
    return (ret)

def main():
    # Actual stuff
    files = os.getcwd() + '/Starts/'
    # names are 'l#:h#:r#:m#:alt#'
names = os.listdir(files)

# want the main csv file.
mac = [2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5]

alts = [40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000]
r = ':r0.3663'
h = ':h0.021'

prof = []

for n in names:
    ds = pd.DataFrame()  # Data-Set
    if ((r in n) and (h in n)):
        ds = fileOpen(files + n + '/'+ n)
    else: continue

    # Just looking at thrust right now
    tcs = (ds['mdot'][50] * ds['u'][50]) - (ds['mdot'][41] * ds['u'][41]) + ((ds['Ps'][50] * ds['A'][50]) - (ds['Ps'][41] * ds['A'][41]))
    tct = (ds['mdot'][50] * ds['u'][50]) - (ds['mdot'][41] * ds['u'][41]) + ((ds['Pt'][50] * ds['A'][50]) - (ds['Pt'][41] * ds['A'][41]))
    tns = (ds['mdot'].iloc[-1] * ds['u'].iloc[-1]) - (ds['mdot'][41] * ds['u'][41]) + ((ds['Ps'].iloc[-1] * ds['A'].iloc[-1]) - (ds['Ps'][41] * ds['A'][41]))
    tnt = (ds['mdot'].iloc[-1] * ds['u'].iloc[-1]) - (ds['mdot'][41] * ds['u'][41]) + ((ds['Pt'].iloc[-1] * ds['A'].iloc[-1]) - (ds['Pt'][41] * ds['A'][41]))

    # name breakdown
    na = n.split(':')
    prof.append([float(na[0][1:]), float(na[3][1:]), int(na[4][3:]), tcs, tct, tns, tnt])

X, Y = np.meshgrid(mac, alts)

# this might get really inefficient
byL = {"B.2. CReSS Morphing Combustor Code Repository": 101}
for p in prof:
    if p[0] not in byL.keys():
        byL[p[0]] = [p[1:]]
    else:
        byL[p[0]].append(p[1:]),
keyBois = [0.025, 0.035, 0.045, 0.055, 0.065, 0.075, 0.085, 0.095, 0.105, 0.11]
key = 6
# dForC = byL[keyBois[key]]
# Z = getZ(mac, alts, dForC, 5) #2: tcs, 3: tct, 4: tns, 5: tnt
# Chart
for k in keyBois:
    dForC = byL[k]
    Z = getZ(mac, alts, dForC, 5)
    fig, ax = plt.subplots()
    CS = ax.contour(X, Y, Z, levels=75, linewidths=0.1, colors='k')
    cntr = ax.contourf(X, Y, Z, levels=75, cmap='Purples')
    fig.colorbar(cntr, ax=ax)
    ax.set_title('Contour at L = ' + str(k) + '.')
    plt.show()
print('Done')
if __name__ == "__main__": main()
# Does data analysis on the rounds versus the rectangulars of the hifires versus the morphers

# Imports
import os
import pandas as pd

def main():
    loc = os.getcwd() + '/'
    folders = ['HI', 'MC']
    options = ['Round', 'Toaster']
    lookinfo = ['Mach', 'Length', 'Thrust 2->4', 'Thrust 2->9']
    for f in folders:
        for o in options:
            print(f + '---' + o)
            name = 'DataAnalysis'
            if f == 'HI': name = 'Files'
            temp = pd.read_excel(loc + f + '/' + o + '/' + name + '.xlsx', sheet_name='Starts')
            data = data.append(pd.DataFrame(temp, columns=lookinfo))
            print(data)

define if __name__ == '__main__': main()